PROYECTO INTEGRADOR DE LA CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

IMPLEMENTACIÓN DE PROTOCOLO DE COMUNICACIONES PARA IOT

Lodovico Molina, Ivo José Martín
Ingeniería

Mg. Guillermo Güichal
Director

Miembros del Jurado
Ing. Roberto Costantini (INVAP - IB)
Ing. Leonardo Brocca (INVAP)

Diciembre de 2018

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
San Carlos de Bariloche, Río Negro, República Argentina

Resumen

En el proyecto se analizaron diferentes protocolos de comunicaciones utilizados para Internet de las Cosas o IoT (por sus siglas en inglés). En base a este análisis se seleccionó a DASH7 Alliance Protocol (D7A) para su implementación. Se evaluó la factibilidad del proyecto y se fijó el alcance del mismo. A continuación, se realizó el diseño y simulación de la capa física de un transmisor y un receptor para FPGA utilizando la tecnología de Síntesis de Alto Nivel o HLS (por sus siglas en inglés). Se llevaron a cabo las pruebas funcionales de cada módulo individual y del comportamiento del sistema global. Por último, el diseño se validó realizando diferentes simulaciones con ruido gaussiano blanco aditivo a la entrada del receptor.

Palabras clave: SÍNTESIS DE ALTO NIVEL, INTERNET DE LAS COSAS, PROTOCOLO DE DASH7 ALLIANCE, PROTOCOLO DE COMUNICACIONES, FPGA

Abstract

In the project, different communication protocols used for the Internet of Things (IoT) were analyzed. Based on this analysis, DASH7 Alliance Protocol (D7A) was selected for its implementation. The feasibility of the project was evaluated and the scope of the project was established. Then, the design and simulation of the physical layer of a transmitter and receiver for FPGA was carried out using the technology of High Level Synthesis (HLS). The functional tests of each individual module and the behavior of the overall system were carried out. Finally, the design was validated by performing different simulations with additive white gaussian noise at the receiver input.

Keywords: HIGH LEVEL SYNTHESIS, INTERNET OF THINGS, DASH7 ALLIANCE PROTOCOL, COMMUNICATIONS PROTOCOL, FPGA

Índice de nomenclatura

3GPP	3rd Generation Partnership Project
AGC	Automatic Gain Control - Contro Automático de Ganancia
ASIC	Application-Specific Integrated Circuit - Circuito Integrado de Aplicación Específica
AXI	Advanced Extensible Interface - Interfaz Extensible Avanzada
BPSK	Binary Phase Shift Keying - Modulación por desplazamiento de fase binario
CS	Coding Scheme - Esquema de codificación
CSS	Chirp spread spectrum - Modulación por Chirp de Espectro Ensanchado
D7A	DASH7 Alliance Protocol - Protocolo de DASH7 Alliance
ENACOM	Ente Nacional de Comunicaciones
FEC	Forward Error Correction - Corrección de Errores hacia Adelante
FIFO	First In First Out - Cola de elementos
FIR	Finite Impulse Response Filter - Filtro con Respuesta finita al impulso
FPGA	Field-Programmable Gate Array - Matriz de Compuertas Programables
FSK	Frequency Shift Keying - Modulación por desplazamiento de frecuencia
GFSK	Gaussian Frequency Shift Keying - Modulación por desplazamiento de frecuencia gaussiano
GMSK	Gaussian minimum shift keying - Modulación por desplazamiento mínimo gaussiano
HLS	High-Level Synthesis - Síntesis de Alto Nivel
I	In-phase component - Componente en fase
IIR	Infinite Impulse Response Filter - Filtro con Respuesta infinita al impulso
IoT	Internet of Things - Internet de las Cosas
ISI	Intersymbol Interference - Interferencia Intersímbolo
ISM	Industrial, Scientific, and Medical radio band - Banda de radio de uso Industrial, Científica y Médico
LFSR	Linear Feedback Shift Register - Registro de Desplazamiento con Retroalimentación

LTE-M	Long Term Evolution for Machines - Evolución a largo plazo para máquinas
LUT	Lookup Table - Tabla de búsqueda
NCO	Numerically Controlled Oscillator - Oscilador Controlado Numéricamente
NFC	Near field communication - Comunicación de campo cercano
OSI	Open Systems Interconnection model - Modelo de Interconexión de Sistemas Abierto
PLL	Phase Locked Loop - Lazo de Seguimiento de Fase
Q	Quadrature component - Componente en cuadratura
QAM	Quadrature Amplitude Modulation - Modulación de amplitud en cuadratura
QPSK	Quadrature Phase Shift Keying
RFID	Radio Frequency Identification - Identificación por radiofrecuencia
RTL	Register Transfer Language - Lenguaje de Transferencia de Registros
RUF	Reservado para Uso Futuro
SIG	Special Interest Group - Grupo de interés especial
SNR	Signal-to-Noise Ratio - Relación Señal a Ruido
TVWS	Television White Space - Espacios libres en bandas de frecuencia de televisión.
UIT	Unión Internacional de Telecomunicaciones
WLAN	Wireless Local Area Network - Red inalámbrica de área local
WPAN	Wireless Personal Area Network - Red inalámbrica de área personal
WWAN	Wireless Wide Area Network - Red inalámbrica de área amplia

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En el siguiente apartado se explica los aspectos fundamentales del proyecto, tales como la motivación y los objetivos del mismo. Por otro lado, se introduce a los conceptos más relevantes para el trabajo como son la Síntesis de Alto Nivel (High-Level Synthesis) y el paradigma de Internet de las Cosas (Internet of Things). Por último se discute acerca de trabajos previos similares.

1.1. Motivación

En los últimos años la forma en que las personas utilizan Internet ha sufrido innumerables transformaciones, afectando todos los aspectos de nuestra vida. En la era de las comunicaciones ubicuas, el foco ha cambiado a la integración de las personas con los dispositivos, en todos los ámbitos. En particular, hubo una explosión en la utilización de dispositivos remotos bajo el concepto de Internet de las Cosas. Este paradigma requiere de comunicaciones que soporten gran cantidad de dispositivos inalámbricos de bajo consumo, conectados en grandes distancias. Atendiendo esta necesidad, se han desarrollado tecnologías muy heterogéneas en cuanto al punto de vista técnico.

Bajo este panorama, han aparecido una variedad de protocolos orientados a estas aplicaciones, tanto de proveedores de telefonía tradicionales (Protocolos 3GPP NB-IoT[1], etc.) como de alternativas independientes (LoRa WAN[2], SigFox[3], etc.). La gran variedad de necesidades y aplicaciones hace que estos protocolos tengan diferentes cualidades y restricciones a la hora de ser implementados.

La heterogeneidad de requerimientos y soluciones plantea un escenario en el que las implementaciones deben ser fácilmente adaptables. Una de las formas de prototipado más comunes es mediante el uso de FPGA (Field-Programmable Gate Array) o dispositivos lógicos configurables. Ambos son ideales para el diseño y prueba de concepto de prototipos, ya que generalmente incluyen recursos de procesamiento de señales digitales y microprocesadores integrados que facilitan el desarrollo.

Por otro lado, la tendencia actual de los proveedores de dispositivos FPGA es la utilización de lenguajes de alto nivel para el diseño de hardware. Estas tecnologías se denominan Síntesis de Alto Nivel o High Level Synthesis (HLS). La conjugación de estas herramientas y su aplicación a las comunicaciones IoT abre el campo a áreas de desarrollo profesional y de investigación modernas con gran potencial de crecimiento en el futuro próximo.

1.2. Objetivos

El proyecto integrador se planteó con los siguientes objetivos:

- Realizar evaluación de protocolos de comunicación actuales y en desarrollo para IoT.
- Analizar la factibilidad del proyecto.
- Proponer un diseño de implementación realizable para un protocolo de comunicaciones de IoT.
- Desarrollar prototipo funcional acorde a la propuesta planteada.
- Presentar documentación técnica del sistema realizado.
- Aportar conceptos para el desarrollo de futuros proyectos integradores o tesis de maestría.

1.3. Síntesis de Alto Nivel o High-Level Synthesis

La Síntesis de Alto Nivel o High-Level Synthesis (HLS) es un proceso de diseño automatizado que interpreta la descripción de un algoritmo realizado en un lenguaje de alto nivel y lo transforma en un sistema digital que responde a ese comportamiento[4]. El método de generación tiene tres fases o etapas principales que se discutirán más adelante. La primera consiste en la especificación de alto nivel y pruebas funcionales, luego se sigue con la síntesis del diseño y termina en la verificación del mismo.

Antes de continuar con el análisis del proceso, se abordará superficialmente el marco histórico en el que está inmerso la tecnología. No es una novedad que la gran capacidad de las tecnologías actuales a base de silicio y la creciente complejidad de las aplicaciones en las últimas décadas han producido un fuerte impacto en la metodologías de desarrollo. En los últimos años se ha iniciado una inclinación por herramientas de desarrollo que poseen un nivel de abstracción más alto para simplificar la construcción de soluciones tecnológicas.

En el contexto del software, por ejemplo, hubo un florecimiento muy marcado. Al comienzo de todo, el lenguaje de máquina era la única forma de programar una computadora, con toda la complejidad que eso implicaba. Al pasar los años se introdujeron metodologías y lenguajes cada vez más abstractos y simples de entender para los programadores. Este cambio de paradigma estuvo de la mano de la abrupta evolución de la tecnología y las arquitecturas de las computadoras, e introdujo simetrías en la sintaxis y semántica de la programación con respecto a nuestra forma procedimental de pensar.

Sin adentrarse en un dominio más filosófico, en el contexto del diseño digital ha sucedido algo análogo. Al principio, los desarrollos eran muy artesanales y las optimizaciones muy difíciles de realizar. A partir de eso se adoptaron métodos de simulación más avanzados, análisis de temporizado y verificación formal de los circuitos los cuales mejoraron las formas de desarrollo. En los años siguientes se introdujeron lenguajes de descripción de hardware tales como VHDL y Verilog. A finales del siglo pasado aparecieron las primeras herramientas comerciales de HLS. A partir de ahí, la complejidad de los sistemas, la multitud de componentes interactuando, la variedad de versiones de los chips y la independencia de los proveedores de componentes hizo que el mercado se enfocara en la reusabilidad e interoperabilidad tanto del hardware como del software. Bajo estas condiciones el uso de HLS se fortaleció. En principio, HLS reduce la cantidad de tiempo necesario para prototipar, crear y verificar el hardware y ajustarlo a una amplia gama de tecnologías como son las FPGA (Field-programmable gate array) o los Circuitos Integrados de Aplicación Específica (Application-Specific Integrated Circuit o ASIC).

El proceso de desarrollo de una aplicación típicamente tiene tres etapas básicas como se dijo anteriormente. El primer paso es llevar a cabo una especificación funcional del sistema utilizando un algoritmo escrito en un lenguaje de alto nivel tal como C/C++ o SystemC. A partir de esto se realiza una simulación de comportamiento para verificar la lógica del algoritmo. Esto quiere decir que se evalúa a priori que ante determinadas entradas de la aplicación, las salidas sean las esperadas, sin tener en cuenta el temporizado de las mismas ni otro tipo de detalle. Esto se lleva a cabo utilizando un compilador convencional del lenguaje de alto nivel elegido, y ejecutando el programa diseñado para verificar su comportamiento. Este tipo de simulación toma todos los datos de entrada y devuelve las salidas de forma simultánea. En este nivel de abstracción, las variables (estructuras y matrices) y los tipos de datos (normalmente, punto flotante y entero) no se relacionan directamente con el dominio de diseño de hardware (bits y vectores de bits). Por lo tanto, la implementación realista de la aplicación requiere una conversión aceptable de tipos de datos enteros y de punto flotante en tipos de datos con precisión de bits de longitud específica (no es un byte o tamaño de palabra estándar, como en el software), mientras se genera una arquitectura de hardware optimizada a partir de esta especificación de precisión de bits.

Una vez que las pruebas de comportamiento fueron realizadas, la herramienta de HLS transforma la especificación funcional de alto nivel en una implementación en lenguaje de descripción de hardware completamente cronometrada. Este proceso de conversión es lo que se denomina como Síntesis, o su traducción en inglés Synthesis. Esta arquitectura generada intenta implementar la especificación de la forma más eficiente posible y además administra los bloques de memoria y las interfaces de comunicación del sistema. Esta descripción en Lenguaje de Transferencia de Registros o Register Transfer Language (RTL) contiene un controlador y el flujo de los datos a través de registros, multiplexores y buses, de forma que se ajusten a la especificación y restricciones de diseño. Los pasos específicos que incluye la síntesis se observan en la Figura 1.1 y se describen a continuación.

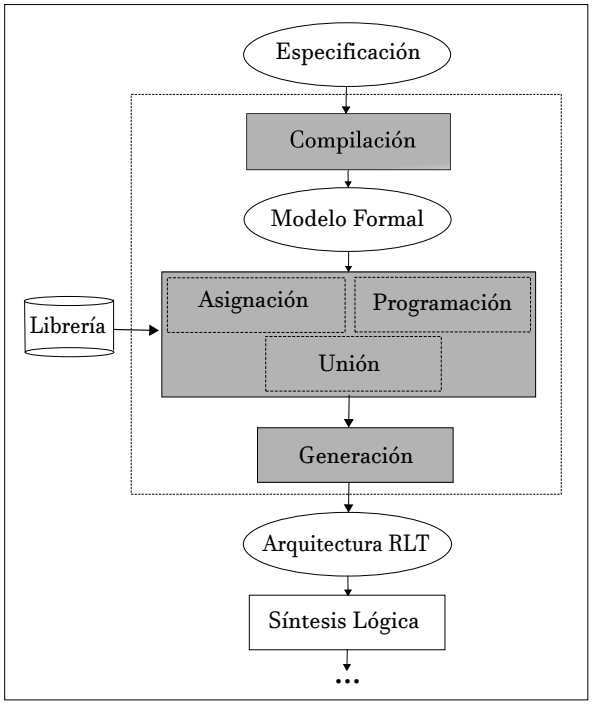


Figura 1.1: Etapas de diseño utilizando Síntesis de Alto Nivel o HLS[4].

- *Compilación y modelado (Compilation and modeling)*: transforma la descripción en una representación formal. Este paso incluye optimizaciones a nivel de código como la eliminaciones de funciones no utilizadas, transformaciones de bucles, etc. El modelo formal resultante tiene como objetivo identificar las dependencia entre las operaciones realizadas y los datos obtenidos en las mismas.

- *Asignación (Allocation)*: define el tipo y la cantidad de recursos necesarios para satisfacer las restricciones del diseño.

- *Programación (Scheduling)*: determina qué operación del algoritmo se ejecuta en cada ciclo de reloj, basándose en la frecuencia del reloj, el tiempo que toma la operación en completarse y las directivas especificadas por el usuario.

- *Unión (Binding)*: determina qué recurso de hardware implementa cada operación programada.
- *Generación (Generation)*: genera el diseño RTL con toda la especificación del hardware necesaria, tomando en cuenta el resultado obtenido de los pasos anteriores.

La especificación que toma el proceso de síntesis no debe contemplar los detalles de implementación a bajo nivel, sino que se deben usar los recursos de HLS para proporcionar ese tipo de restricciones. Esta especificación debe ser pensada de forma secuencial y no incluir nociones de paralelismo. Siempre se debe mantener la abstracción porque de otra forma se limita el diseño que HLS puede generar. Para gestionar este tipo de detalles HLS cuenta con un conjunto de directivas y pragmas que especifican los objetivos a bajo nivel que debe cumplir el diseño, como pueden ser el tipo de interfaces, tipo de paralelismo requerido, forma de tubería o pipeline, restricciones de reloj, tipos de recursos de hardware usados y demás.

El último paso del proceso consiste en la Verificación o Co-simulación. El mismo consiste en introducir las entradas descriptas en los casos de prueba funcionales del comienzo al diseño RTL y verificar que las salidas sean las mismas que las que se esperaban de la descripción funcional. Todo este proceso es realizado automáticamente por la herramienta de desarrollo. Esta etapa sirve para corroborar que el diseño generado tenga el mismo comportamiento que se esperaba del código escrito en alto nivel. Adicionalmente se puede observar un diagrama de tiempos que muestra en el dominio de los ciclos de reloj, cuándo se ingresan las entradas y cómo evolucionan las salidas a través del tiempo.

Al final de todas estas etapas se puede realizar un empaquetado del diseño de RTL para luego utilizarlo en una herramienta que permita el análisis y la síntesis de diseños en lenguaje de descripción de hardware.

1.4. Internet de las Cosas o Internet of Things

No existe una única definición del término Internet de las Cosas o Internet of Things (IoT) que sea aceptada por todos los usuarios a nivel mundial. Se considera que la primera vez que se utilizó el término fue realizada por Kevin Ashton en una presentación donde relacionó la tecnología de RFID y la gestión en la cadena de suministros[5]. Desde entonces, el término fue definido de muchas formas. Según la Unión Internacional de Telecomunicaciones (UIT)[6], es una infraestructura mundial para la sociedad de la información que propicia la prestación de servicios avanzados mediante la interconexión de objetos (físicos y virtuales) gracias a la interoperatividad de tecnologías de la información y la comunicación presentes y futuras. IoT no es una sola tecnología,

sino que más bien es una aglomeración de varias tecnologías que trabajan juntas en conjunto.

Según la UIT, las características fundamentales de IoT son las siguientes[6]:

- *Interconectividad*: en el contexto de IoT, todo puede estar interconectado con la infraestructura mundial de la información y la comunicación.
- *Servicios relacionados con objetos*: IoT es capaz de suministrar servicios relacionados con los objetos dentro del mundo físico o virtual.
- *Heterogeneidad*: los dispositivos en IoT son heterogéneos dado que se basan en diferentes plataformas hardware y redes. Pueden interactuar con otros dispositivos o plataformas de servicios a través de redes diferentes.
- *Cambios dinámicos*: el estado de los dispositivos varía dinámicamente, por ejemplo del modo reposo al activo, conectado y/o desconectado, así como el contexto del dispositivo, como la ubicación y velocidad. Además, el número de dispositivos también puede cambiar dinámicamente.
- *Escala enorme*: el número de dispositivos que ha de gestionarse y que se comunican entre sí puede ser incluso un orden de magnitud mayor que el número de dispositivos conectados actualmente a Internet. El porcentaje de comunicación que requerirán estos dispositivos será muchísimo mayor que el de la comunicación entre humanos. Será incluso más esencial la gestión de los datos generados y su interpretación para fines de aplicación, aspectos éstos que guardan relación con la semántica de datos y la manipulación eficiente de datos.

1.5. Metodología de trabajo

En primer lugar, los esfuerzos se centraron en la investigación e indagación sobre los temas que abarca el proyecto. En particular se realizó una lectura exploratoria sobre los diferentes protocolos abarcados por IoT. Se evaluaron diferentes factores de cada uno de ellos para poder compararlos en más de un aspecto y poder elegir uno de manera fundamentada.

A partir de literatura oficial de cada uno de los protocolos y de trabajos independientes, se resumió de forma compacta las características más importantes de los mismos. Con esa información se eligió uno de ellos utilizando un criterio explicado en el apartado correspondiente.

El siguiente paso consistió en un estudio de factibilidad para verificar que el proyecto sea posible en términos técnicos, legales y económicos. En el aspecto técnico se evaluó que las tecnologías disponibles para el proyecto alcanzaran para llevarlo a cabo de

forma correcta. En el ámbito legal se realizó una investigación sobre las restricciones que existen sobre este tipo de implementaciones. En particular se indagó sobre las resoluciones correspondientes del Ente Nacional de Comunicaciones (ENACOM) para poder tomar una postura informada. En el aspecto económico o financiero se analizó de forma simple si el proyecto tenía algún problema de carácter monetario.

Una vez concretados los pasos anteriores, la implementación del protocolo elegido comenzó con un diseño conceptual de la cadena completa de transmisión y recepción. Dado que el tiempo disponible para la realización del proyecto fue acotado, se definió al comienzo del mismo el alcance que iba a tener el trabajo en cuanto a la implementación y pruebas. Luego, a partir de las especificaciones técnicas del protocolo se elaboró un modelo de transmisor y uno de receptor que se ajustara a las mismas. El resultado de este proceso fueron dos diagramas en bloque de los elementos necesarios para implementar la cadena completa de comunicación abarcada por el alcance.

Con el modelo conceptual en mente, se implementó y probó cada bloque por separado. El desarrollo se realizó con HLS utilizando la metodología de trabajo descripta anteriormente. Cada vez que la verificación era exitosa (es decir, arrojaba el resultado esperado), se incorporó el bloque a la cadena de transmisión o recepción.

Una vez finalizados ambos sistemas, se realizó pruebas de integración para verificar que tanto el transmisor como el receptor funcionen de manera correcta en conjunto. Por último se simuló el comportamiento del receptor en presencia de ruido gaussiano blanco aditivo.

1.6. Trabajos previos

A partir de la búsqueda de trabajos de investigación que involucraran estos temas, se encontraron muchos artículos relacionados tanto a IoT como a HLS, pero ninguno que vinculara de forma directa y técnica ambos temas. Se trata de dos paradigmas que en principio poco tienen que ver y cuya unión no resulta trivial. Se leyeron algunos artículos que proponen el uso de HLS para el prototipado de dispositivos IoT debido a la velocidad de desarrollo y facilidad en la verificación del diseño. Sin embargo no se encontró ningún trabajo que validara esto en un caso real. Si bien HLS se ha utilizado con éxito en protocolos de alto nivel, como por ejemplo protocolos de transmisión de video, no hay documentación sobre su uso en algún protocolo de comunicaciones de IoT.

Capítulo 2

Análisis de protocolos de IoT

En el siguiente apartado se realiza una descripción superficial de los protocolos evaluados en esa etapa del proyecto. Se explican los criterios que se tuvieron en cuenta para el análisis. A continuación se amplía en los aspectos más fundamentales de cada protocolo en base a los criterios elegidos. Por último se muestra una tabla comparativa que presenta de forma compacta las diferencias entre cada uno de ellos.

2.1. Criterios de evaluación

Existe una extensa cantidad de estándares e iniciativas de tecnologías inalámbricas muy diferentes entre ellas utilizadas en IoT. Para empezar a compararlas primero se deben poner en la mesa ciertos puntos a tener en cuenta sobre cada uno de ellos. En el siguiente listado se muestran los aspectos que se tuvieron en cuenta para su análisis:

- *Rango*: hace referencia a la distancia a la que los nodos pueden comunicarse en la red.
- *Tasa de datos*: no es un detalle menor ya que limita la aplicación sobre la red.
- *Consumo de potencia*: es un aspecto importante a la hora de hacer una diferencia entre protocolos siendo una pieza clave en IoT.
- *Frecuencia de operación*: cada protocolo está desarrollado pensando en frecuencias de portadoras distintas de acuerdo a las normativas de cada lugar y las aplicaciones de cada uno de ellos. Existen protocolos que utilizan bandas del espectro no licenciadas y otros protocolos que no[7].

Los aspectos de rango de alcance, la tasa de transmisión y el consumo de potencia están íntimamente relacionados. No existe un protocolo ni una tecnología que logre destacar en todos los aspectos simultáneamente, es por esto que fue necesario priorizar

de cierta forma algún aspecto a la hora de elegir un protocolo. Se decidió inclinarse hacia aquellos que posean mayor rango. Podría argumentarse que un protocolo de estas características es conveniente para soluciones que requieran amplias zonas cubiertas de nodos, como podría pasar en un país como el nuestro. De todas formas, como no se espera tener un prototipo completo luego del proyecto para satisfacer alguna necesidad en particular, este argumento es simplemente una inclinación para tomar una decisión no arbitraria con un fin académico.

Otra característica fundamental que se buscó fue que el protocolo opere en bandas de frecuencia no licenciadas. Esto se debe a que resulta mucho más sencillo llevar al mercado dispositivos con esta característica ya que sólo deben ser homologados mediante la norma técnica correspondiente[8].

Además de lo anterior, la lista de protocolos fue filtrándose por si misma debido a las limitaciones o restricciones en su implementación que presentaban cada uno de ellos, como por ejemplo la disponibilidad de su especificación técnica.

2.2. Descripción de protocolos

En primer lugar, los protocolos se clasifican de acuerdo a su rango de operación y se analizan los más importantes de cada clasificación. En la Figura 2.1 se observa una serie de protocolos relevantes clasificados según su alcance[9].

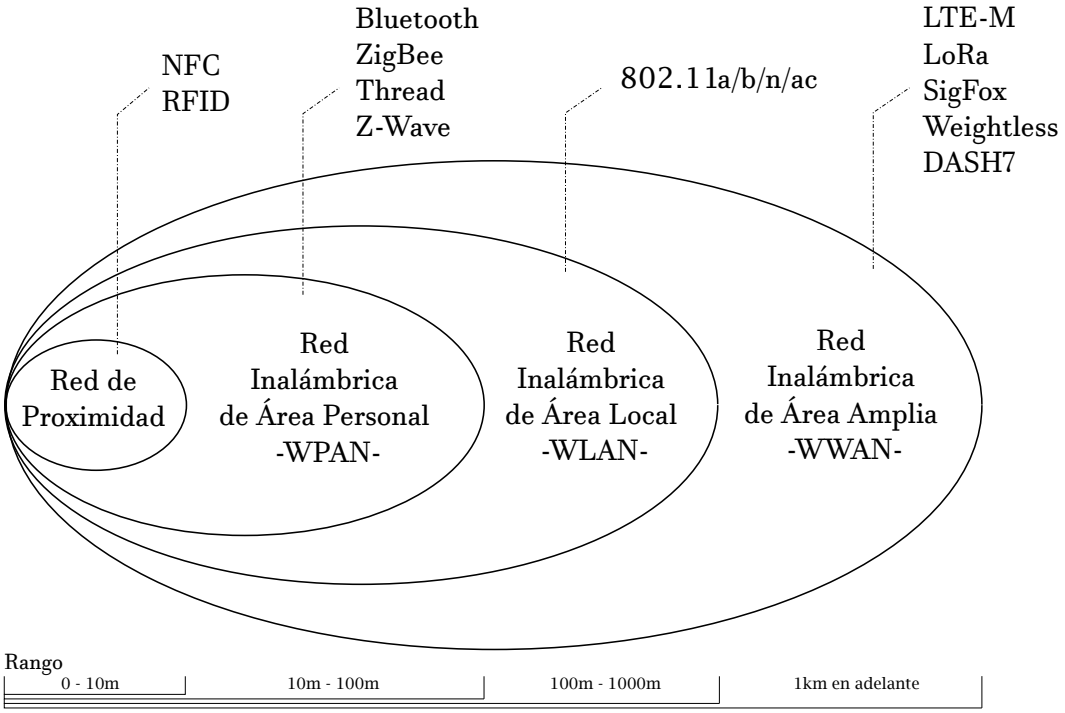


Figura 2.1: Tecnologías más relevantes de conectividad inalámbrica para IoT clasificadas según su rango[9].

A partir de una lectura exploratoria de varios artículos sobre estos protocolos, se

pudo observar marcadas tendencias que hacen a algunos de ellos imponerse fuertemente sobre los otros. Se analizan estos protocolos en mayor profundidad [10].

2.2.1. Protocolos de Corto Alcance

Este tipo de protocolos posee un rango de operación desde algunos centímetros hasta decenas de metros. No se dedicó demasiados esfuerzos en esta clasificación de protocolos por lo que se explicó anteriormente. Sin embargo se los llegó a conocer por ser tecnologías de IoT y se los incluye en este informe de forma compacta por cuestiones de completitud. Los principales protocolos de IoT que entran dentro de esta clasificación son los se muestran a continuación.

NFC

Permite una comunicación entre dos dispositivos manteniéndolos a una distancia entre ellos de típicamente 10 cm. Posee una frecuencia de portadora de 13,56 MHz y una tasa de datos máxima de 424 kbps. Este protocolo es usado principalmente por dispositivos móviles como celulares.

Bluetooth Smart o Bluetooth de bajo consumo

Manteniendo un rango similar al Bluetooth convencional, es decir algunas decenas de metros, reduce su consumo de energía disminuyendo la tasa de datos transmitida. Utiliza la banda no licenciada de 2,4 GHz y tiene una tasa de transferencia máxima de 1 Mbps. Todos los dispositivos que certifiquen el estándar 4.1 de Bluetooth son compatibles con esta variante también.

Z-Wave

Se trata de un protocolo inspirado en la automatización del hogar. Trabaja sobre una frecuencia de 868 kHz y una tasa de transmisión máxima de 100 kbps. A su vez posee un alcance de 30 m en ambientes de interior y hasta 100 m en el exterior.

ZigBee

Se trata de un estándar que soporta varias topologías de red (ya sea estrella, árbol o malla) y desarrollado para uso personal como industrial creando redes locales de bajo consumo. Funciona en banda no licenciada de 2,4 GHz o en 868 MHz. Se utiliza para redes de corto alcance ya sea algunas decenas de metros para ambientes de interior o hasta 100 m en el exterior. Además posee una tasa de datos máxima de 250 kbps.

Thread

Se trata de un estándar pensado para dispositivos hogareños basado en IPv6. Opera en una frecuencia de 2,4 GHz y soporta una tasa de datos de hasta 250 kbps en un rango de 30 m. Utiliza una topología de malla y se basa en el estándar IEEE 802.15.4 para comunicaciones inalámbricas de bajo consumo.

2.2.2. Protocolos de Medio y Largo Alcance

Son tecnologías que tienen un alcance que están en el rango de los cientos de metros hasta varios kilómetro, incluso algunos de ellos pueden llegar a operar hasta los 15 km en espacios abiertos. Los protocolos más importantes de esta clasificación son los que se mencionan a continuación.

LTE-M

Esta tecnología fue una de las propuestas del grupo 3GPP (colaboración de grupos de asociaciones de telecomunicaciones, conocidos como miembros organizativos) para el crecimiento de las redes IoT[1].

Se trata de un estándar de poca complejidad y baja tasa de datos y pensado para ser desarrollado en frecuencias menores a 1 GHz. Se optimizó para comunicaciones entre dispositivos con bajo consumo, pero sin embargo no llega a ser tan de bajo consumo como los siguientes candidatos.

Existen versiones del estándar como puede ser 3GPP que soporta una tasa de datos máxima de 1 Mbps.

Otra variante por ejemplo es también NB-LTE-M la cual disminuye el ancho de banda de 1,4 MHz a 0,2 MHz pero aumenta el alcance hasta 15 km.

Una de las principales ventajas es que utiliza las redes de celulares ya existentes, por lo que no es necesario mayores implementaciones. Aunque esto también trae su desventaja, ya que el soporte está sujeto a la disponibilidad y cobertura del proveedor.

Este protocolo opera en bandas de frecuencia licenciadas, por lo que no es la mejor opción para un trabajo académico de este tipo.

LoRa

Es un estándar iniciado por SemTech y hoy en día está bajo la tutela de LoRa Alliance. Haciendo referencia al famoso modelo de capas que se conoce en redes, las capas inferiores de este estándar son desarrolladas directamente en placas de SemTech (protocolo propietario) y las capas superiores del estándar son abiertas. Esta limitación hace que no sea factible el desarrollo de un prototipo de las capas inferiores.

Esta tecnología usa modulación de espectro ensanchado en bandas no licenciadas menores a 1 GHz. Normalmente se conoce su modulación como "Chirp de Frecuencia Modulada", con una patente del circuito bajo su poder. Otras formas de modular que tiene su estándar son FSK y PSK, dependiendo del dispositivo adquirido[2].

Su alcance varía entre 5 km a 15 km dependiendo si se encuentra en zonas urbanas o no respectivamente y ofrece tasas de transmisión entre 0,98 kbps a 21,9 kbps.

Actualmente se encuentra muy fuerte en el mercado debido a su bajo costo, escalabilidad y desempeño.

Sigfox

Se trata de una tecnología de origen francesa muy establecida en el mercado. La misma consiste en redes celulares desplegadas principalmente en Europa que le dan soporte a dispositivos de muy bajo consumo y muy económicos[3].

Utiliza modulación de Banda Ultra Angosta o Ultra Narrow Band (UNB) en zonas no licenciadas del espectro menores a 1 GHz. Utiliza una transmisión a tasas muy bajas como 100 bps pero a distancias muy grandes del orden de 10 km en áreas urbanas y 50 km en áreas rurales. Se trata de un estándar propietario también.

Una de las principales ventajas de este tipo de tecnología es que no se requiere realizar despliegues de redes troncales que conecten los nodos a internet. Esto ya se encuentra solucionado por el proveedor de la cobertura móvil necesaria para que los equipos funcionen.

Por otro lado se trata de un protocolo privativo, por lo que sus especificaciones técnicas no son liberadas a la comunidad. Esto imposibilita una implementación independiente.

Weightless

Se trata de un conjunto de estándares con diferentes finalidades de bajo consumo dedicados a IoT con topología de estrella [11].

En primer lugar Weightless-N es un sistema de ancho de Banda Ultra Angosta o Ultra Narrow Band (UNB) similar en varios aspectos a Sigfox, enfocado en subida de datos desde sensores a una tasa de transmisión muy baja (alrededor de 100 bps) en bandas no licenciadas con alcance de hasta 2 km.

Por otro lado Weightless-W también pertenece a este conjunto, con la diferencia que utiliza el espectro de los espacios blancos del espectro de TV en la banda UHF, es decir a frecuencias mucho mayores a la de los protocolos anteriores. Por otro lado permite una velocidad de transmisión de hasta 10Mbps en distancias de hasta 5 km. Esta variante también requiere un consumo mayor de energía que la anterior.

La variante más reciente se trata de Weightless-P. La misma ofrece una comunicación bidireccional como la variante W pero con ancho de banda angosto como la versión N. Permite por su lado un alcance de 2 km y una tasa de transmisión tanto de subida como de bajada de 200 bps.

Weightless es promocionado como un un estándar abierto aunque sólo es compartido si uno es miembro de Weightless SIG, la institución que soporta el protocolo.

DASH7 Alliance Protocol (D7A)

Se trata de un estándar completamente libre y abierto. El mismo es soportado por miembros de DASH7 Alliance, en los que se encuentran empresas de renombre internacional como Cortus, Institute of Logistics and Warehousing, Kawantech, Neotion, entre otros, y universidades importantes de todo el mundo como Pusan University (Corea), University of Antwerp (Bélgica), University of Bremen (Alemania) y Wroclaw University of Technology (Polonia) [12].

El protocolo opera en bandas no licenciadas al igual que varios de los otros candidatos, con tasa de transmisión variable del orden de los 9kbps hasta los 166kbps. Tiene un alcance de hasta 5km, aunque se recomienda enlaces con rango menor a 1 km. Además soporta todas las topologías de red (es decir malla, árbol y estrella) con transmisión multi salto de paquetes.

Éste fue el único candidato que presenta todas las condiciones favorables. Es decir, su especificación es abierta y opera en bandas no licenciadas, por lo que resultó ser la opción más factible de implementar. Si bien no es un protocolo muy conocido ni utilizado, para un trabajo académico de este estilo tiene todas las cualidades que se buscaban.

2.3. Tabla comparativa

A continuación se muestra una tabla comparativa de las tecnologías descriptas anteriormente de las características de cada una. En la misma se presentan estas características que se consideran decisivas a la hora de elegir un protocolo.

	LTE-M	LoRa	Sigfox	Weightless-N	Weightless-W	Weightless-P	D7A
Banda de frecuencia	Celular	Sub 1 GHz ISM	Sub 1 GHz ISM	Sub 1 GHz ISM	TVWS (UHF)[1]	Sub 1 GHz ISM	Sub 1 GHz ISM
Ancho de banda (canal)	1,4 MHz	125 kHz	200 Hz	200 Hz	5 MHz	12,5 kHz	9,6 kHz, 100 kHz ó 83 kHz
Rango urbano	2,5 km	2 km	10 km	3 km	5 km	2 km	1 km
Rango rural	5 km	15 km	50 km	—	—	—	5 km
Tasa de datos	200 kbps	1 kbps–22 kbps	0,1 kbps	0,1 kbps	1 kbps–10 Mbps	0,2 kbps–100 kbps	9,6 kbps, 55 kbps ó 166 kbps
Topologia	Estrella	Estrella sobre estrella	Estrella	Estrella	Estrella	Estrella	Malla, Estrella o Árbol
Institución soporte	3GPP	LoRa Alliance	Sigfox	Weightless SIG	Weightless SIG	Weightless SIG	DASH7 Alliance
Licencia	Abierto	Parcialmente abierto	Privativo	Abierto con membresía	Abierto con membresía	Abierto con membresía	Abierto

1 Espacios libres en bandas de frecuencia utilizadas para televisión en la banda UHF.

Tabla 2.1: Comparación de los aspectos más importantes de los protocolos evaluados que se utilizan para IoT de medio y largo alcance.

Capítulo 3

Protocolo DASH7 Alliance

El protocolo de DASH7 Alliance (D7A) es una interfaz aérea inalámbrica, y una pila de sistemas, que opera exclusivamente en las bandas ISM Sub-1 GHz establecida por la UIT[7]. Está diseñado para aplicaciones de muy baja potencia, como RFID activo y redes de sensores inalámbricos. En este capítulo se detallan las características generales del protocolo y las especificaciones técnicas de la capa implementada. El siguiente texto fue extraído de la especificación del protocolo D7A proporcionado por DASH7 Alliance[13], la que se reserva todos los derechos de propiedad intelectual correspondientes.

3.1. D7A: un protocolo de comunicación BLAST

- *En ráfaga (Bursty)*: la transferencia de datos es abrupta y no incluye transmisiones de contenido continuo como video, audio u otras formas de datos isócronas.
- *Ligera (Light)*: el tamaño de los paquetes está limitado a 256 bytes.
- *Asíncrono (Asynchronous)*: el método de comunicación es mediante petición-respuesta, el cual no requiere de establecimiento de comunicación periódico o sincronización entre dispositivos.
- *Sigiloso (Stealth)*: no usa señales balizas de descubrimiento, los nodos terminales pueden elegir responderle solo a dispositivos pre-aprobados.
- *De transición (Transitional)*: un sistema de dispositivos D7A es inherentemente móvil o de transición. A diferencia de otras tecnologías inalámbricas, D7A está centrado en la carga, no está centrado en la descarga. Los dispositivos no necesitan ser administrados extensamente por una infraestructura fija (es decir, estaciones base) para responder solo a los dispositivos pre-aprobados.

3.2. Organización

En las especificaciones del protocolo se realiza la siguiente correspondencia entre D7A y el Modelo de capas OSI:

- *Capa física*: define las características técnicas del espectro, modulación y las características de codificación de canal.
- *Capa de enlace de datos*: define la recepción, transmisión, automatización de escaneo y procesos de accesos múltiples. Provee el primer nivel de filtrado de tramas y el detalle de armado de las mismas. Define el concepto de Perfil de Acceso y Clase de Acceso.
- *Capa de red*: define dos protocolos, uno en segundo plano (D7AAdvP o DASH7 Alliance Advertising Protocol) y otro en primer plano (D7ANP o DASH7 Alliance Network Protocol). D7AAdvP se utiliza para la sincronización ad-hoc de dispositivos que participan en escaneos de ultra bajo consumo de energía. D7ANP proporciona seguridad de red (autenticación y encriptación) y hasta un enrutamiento de salto.
- *Capa de transporte*: define el concepto de petición-respuesta. Define un método para acusar el recibo de peticiones simples y grupales. Provee una serie de herramientas para minimizar el uso de D7AAdvP a través de una extensión del escaneo en primer plano.
- *Capa de sesión*: define el concepto de Calidad de Servicio (conocido como Quality-of-Service) y el método para poner en cola, programar, transmitir, retransmitir y recibir solicitudes de la capa superior.
- *Capa de elementos de datos*: define el sistema de archivos de D7A. Contiene la configuraciones de D7A y los datos altamente estructurados definidos por el usuario. Define atributos de permisos similares a POSIX y atributos de acción específicos de D7A, utilizados para activar acciones de aplicaciones. Define 4 clases de almacenamiento.
- *Interfaz de capa de aplicación*: es la Interfaz de Programación de Aplicaciones (API) de D7A. Define los roles de D7A (usuario, invitado y administrador). Proporciona un direccionamiento sofisticado basado en propiedades.

3.3. Capa física

A continuación se detallan las especificaciones técnicas de la capa física del protocolo. En el documento de la especificación que provee DASH7 Alliance existen detalles

que no están expuestos en los siguientes apartados. Ya que la implementación en un dispositivo físico no es parte del alcance del proyecto, se decidió no incluir las precisiones sobre la etapa de radiofrecuencia (RF) y la señal pasabanda.

3.3.1. Utilización del espectro y canales

En esta sección se encuentra aquello que corresponde a la utilización del espectro y características de cada canal de transmisión.

Bandas de espectro

D7A se puede utilizar en cualquiera de las siguientes bandas de Sub-1 GHz sujeto a la conformidad con las regulaciones locales. La siguiente tabla define los índices y los rangos de frecuencia de las bandas de canales en D7A frente a las bandas ISM Sub-1 GHz:

Banda ISM	Índice de Banda	Inicio de la banda	Fin de la banda
RUF[1]	0x0-0x1	—	—
433 MHz	0x2	433,060 MHz	434,785 MHz
868 MHz	0x3	863 MHz	870 MHz
915 MHz	0x4	902 MHz	928 MHz
RUF[1]	0x5-0x7	—	—

1 Reservado para Uso Futuro.

Tabla 3.1: Bandas de canales en D7A.

Las clases de canal especifican las tasas de datos y los esquemas de modulación para cada canal D7A. Los esquemas de modulación asociados a cada clase de canal se definen de acuerdo con la Tabla 3.2. Es importante aclarar que todos los esquemas utilizan modulación 2-(G)FSK. Si se usa la conformación del pulso gaussiano (por lo tanto, GFSK), el producto ancho de banda-tiempo de bit del filtro gaussiano estará en el rango de 0.3 a 1.0.

3.3.2. Codificación de canal

Los mensajes transmitidos o recibidos a través de las clases de canal D7A están compuestos por símbolos binarios. Los símbolos están codificados por uno de los métodos de codificación de canal admitidos por D7A. Es deber de las capas superiores configurar el codificador y el decodificador según sea necesario, para utilizar el método que sea necesario.

Clase de canal[1]	Índice	Tasa de símbolos	Índice de mod.	Símbolo 0	Símbolo 1
Low-Rate[2]	0x0	9,6 kbps	1	−4,800 kHz	+4,800 kHz
RUF[4]	0x1	—	—	—	—
Normal[3]	0x2	55,555 kbps	1,8	−50 kHz	+50 kHz
Hi-Rate[3]	0x3	166,667 kbps	0,5	−41,667 kHz	+41,667 kHz

1 Todos los esquemas utilizan modulación 2-(G)FSK.
2 Posee un espaciamiento entre canales de 250 kHz.
3 Posee un espaciamiento entre canales de 200 kHz.
4 Reservado para Uso Futuro.

Tabla 3.2: Esquemas de modulación en D7A y sus respectivas características principales.

Ordenamiento de bytes (Endianness)

El contenido de todos los mensajes transmitidos está en formato *Big Endian*, el byte más significativo primero.

Codificación PN9

La codificación PN9 es una codificación estadísticamente sin nivel de continua, de tasa completa, que no ofrece ganancia de codificación. Tanto la codificación como la decodificación requieren el uso de un Registro de Desplazamiento con Retroalimentación (Linear Feedback Shift Register o LFSR) y un polinomio semilla para producir una secuencia predecible de valores pseudoaleatorios. A esta secuencia, junto con el flujo de datos, se les realiza la operación lógica XOR. El polinomio PN9 se inicializa con el siguiente valor: $x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$. El esquema de la codificación PN9 se puede observar en la Figura 3.1.

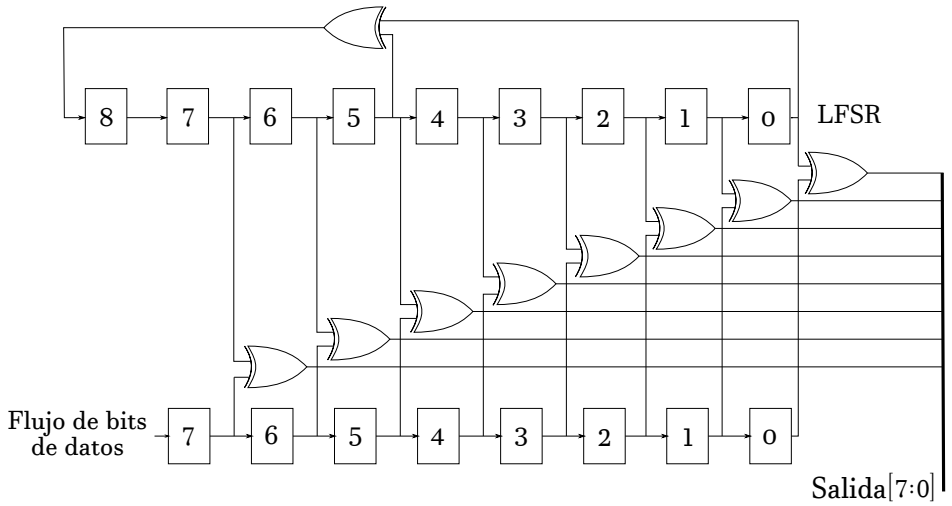


Figura 3.1: Esquema de codificación y decodificación del flujo de datos utilizando PN9.

Codificación FEC

La corrección de errores hacia adelante (Forward Error Correction o FEC) es un mecanismo de detección y corrección de errores. A continuación se explica el método en particular que utiliza el protocolo para implementar esta técnica.

Codificador y decodificador convolucional La primera etapa del codificador y la segunda etapa del decodificador es el cálculo de un código convolucional a partir de los datos no codificados. El código es un código convolucional de tasa 1/2, que utiliza un algoritmo específico de longitud igual a 4. Como la entrada al entrelazador debe estar alineada a 32 bits, la terminación del trellis del código convolucional adjunto a los datos no codificados es `0x0B0B` para una cantidad de bytes par y `0x0B0B0B` para una cantidad de bytes impar. En la siguiente tabla se muestran las salidas en función del bit de entrada b_{in} y de los tres bit anteriores, donde S_1 es el bit más antiguo y S_3 es el más reciente.

S_1	S_2	S_3	b_{in}	**Salida**
0	0	0	0	00
0	0	0	1	11
0	0	1	0	01
0	0	1	1	10
0	1	0	0	11
0	1	0	1	00
0	1	1	0	10
0	1	1	1	01
1	0	0	0	11
1	0	0	1	00
1	0	1	0	10
1	0	1	1	01
1	1	0	0	00
1	1	0	1	11
1	1	1	0	01
1	1	1	1	10

Tabla 3.3: Salida del codificador convolucional en función del bit de entrada y los tres bits anteriores.

Entrelazador La segunda etapa del codificador y la primera etapa del decodificador es un proceso de entrelazado y desentrelazado de matriz, que está diseñado para separar los datos adyacentes para que los errores de datos de ráfagas puedan ser tratados de

manera más efectiva por el corrector de errores de código convolucional. Las operaciones de intercalado y desentrelazado se ejecutan en símbolos de 2 bit. En la Figura 3.2 se puede observar la regla que sigue el entrelazador.

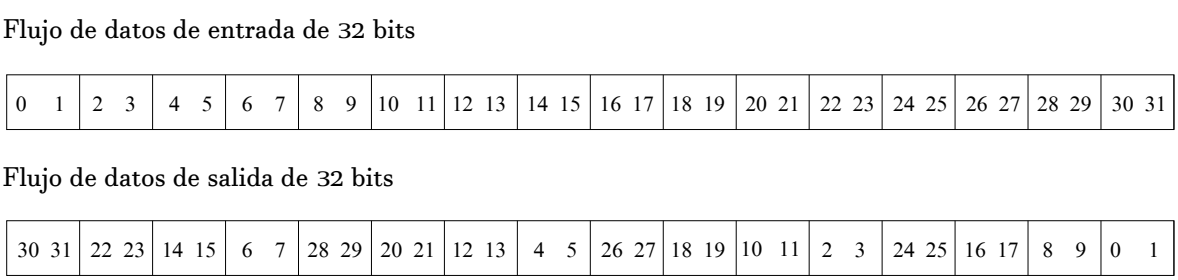


Figura 3.2: Regla de entrelazado aplicada a la salida del código convolucional. Constituye la segunda etapa de en la Codificación FEC.

Esquemas de codificación

La siguiente tabla especifica los índices de los diferentes esquemas de codificación:

Nombre del esquema	Índice	Método de codificación
CS0	0x0	PN9[1]
CS1	0x1	RUF[3]
CS2	0x2	FEC+PN9[2]
CS3	0x3	RUF[3]

1 Los símbolos a transmitir se codifican con PN9 antes de la modulación y la transmisión. Simétricamente, en la recepción, el flujo de datos demodulados se decodifica en primer lugar con PN9 antes de ser procesados.

2 El flujo de símbolos se codifican primero utilizando el FEC de tasa 1/2 y luego con la codificación PN9 antes de la modulación y transmisión. Simétricamente, se decodifica en primer lugar con PN9 y le sigue la decodificación del FEC de tasa 1/2.

3 Reservado para Uso Futuro.

Tabla 3.4: Índices de esquemas de codificación de canales en D7A.

3.3.3. Estructura de paquete

Todo el tráfico de datos en los canales D7A es en forma de paquetes que poseen la estructura que se muestra en la Figura 3.3. Los paquetes D7A contienen un preámbulo, una palabra de sincronismo y una carga útil. El preámbulo es una serie de símbolos binarios alternados que comienzan con '1' y se utilizan para calibrar la tasa de transmisión en el receptor. La palabra de sincronismo es un bloque de 16 símbolos binarios que se utiliza para alinear la carga útil del paquete. La carga útil del paquete contiene

datos codificados. La palabra de sincronismo y la carga útil son temas de secciones posteriores de la especificación.

Además, los períodos de aumento (o rampa ascendente) y disminución de la potencia (o rampa descendente) pueden preceder y seguir el paquete respectivamente. Estos períodos deben ser lo más cortos posible, utilizados con el fin de cumplir con los requisitos de ancho de banda del canal.

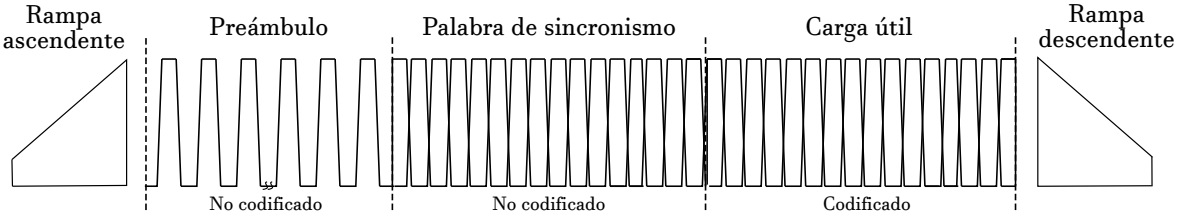


Figura 3.3: Estructura general de todos los paquetes presentes en el protocolo D7A.

En la siguiente tabla se muestra la cantidad de símbolos que ocupa cada parte del paquete en función de las especificaciones.

Especificación	Mínimo	Típico	Máximo
Preámbulo (Clase Low Rate)	32 símbolos	32 símbolos	64 símbolos
Preámbulo (Clase Normal)	32 símbolos	32 símbolos	64 símbolos
Preámbulo (Clase Hi-Rate)	48 símbolos	48 símbolos	128 símbolos
Período de rampa ascendente	0 símbolos	8 símbolos	32 símbolos
Período de rampa descendente	0 símbolos	8 símbolos	32 símbolos
Palabra de sincronismo	16 símbolos	16 símbolos	16 símbolos

Tabla 3.5: Especificaciones de requisitos de paquetes.

Palabra de sincronismo

La capa física admite varias clases de palabras de sincronismo, con el fin de diferenciar los paquetes. La siguiente tabla ofrece una visión general de los valores de las palabras de sincronismo según el esquema de codificación y la clase de palabra de sincronismo.

Clase de palabra de sincronismo	Esquema de codificación			
	CS0	CS1	CS2	CS3
0	0xE6D0	RUF[1]	0xF498	RUF[1]
1	0x0B67	RUF[1]	0x192F	RUF[1]

1 Reservado para Uso Futuro.

Tabla 3.6: Valor de la palabra de sincronismo en D7A en función de la clase de palabra de sincronismo y el esquema de codificación.

Capítulo 4

Estudio de factibilidad

Como en todo proyecto de ingeniería, el estudio de factibilidad representó una parte importante del análisis previo a la implementación. En este capítulo se explica cómo se evaluó la factibilidad técnica, legal y financiera del proyecto para poder corroborar si el mismo era realizable en estos términos. En el aspecto técnico se evaluó que las tecnologías disponibles para el proyecto fueran suficientes para llevarlo a cabo de forma correcta. En el ámbito legal se realizó una investigación sobre las restricciones de la ley sobre este tipo de implementaciones. En el aspecto económico o financiero se analizó de forma simple si el proyecto tenía algún problema de carácter monetario.

4.1. Factibilidad técnica

El proyecto conlleva dos grandes desafíos los cuales se evaluaron al comienzo. Por un lado, diseñar el transmisor y el receptor se consideró una dificultad relevante en el trabajo, sin embargo, en el transcurso de la carrera se obtuvieron herramientas y conocimientos suficientes para afrontar el reto.

El segundo desafío residía en el completo desconocimiento que se tenía de HLS al comenzar el proyecto. No es un tema abarcado en la carrera y además, no existe mucha documentación sobre el paradigma debido a que se comenzó a usar en los últimos años. De todas formas se llegó a la conclusión de que el proceso de trabajo se podía adoptar con un margen de tiempo dedicado al aprendizaje del mismo.

4.2. Factibilidad legal

Dado que el espectro electromagnético es un bien escaso, el ENACOM (Ente Nacional de Comunicaciones) dispone de normas que deben respetar todos los artefactos que hagan uso del mismo. Es por esto que resulta obligatorio verificar que cualquier dispositivo desarrollado se ajuste a las normas dispuestas por el ENACOM.

El protocolo elegido opera en la banda de frecuencia no licenciada comprendida entre los 902–928MHz, por lo que no es necesario solicitar una licencia de uso de espectro al ENACOM según la Resolución N° 226/2008[14], en concordancia con el lineamiento internacional marcado por la UIT (Unión Internacional de Telecomunicaciones). Sin embargo, que no se requiera una licencia no significa que no se deba respetar cierta reglamentación. En particular, los dispositivos de uso comercial deben ser homologados según la norma correspondiente. El ENACOM dispone de ciertas restricciones técnicas que se deben atender para realizar la homologación de un dispositivo en esta banda, las cuales están listadas más adelante.

Las normas técnicas para la homologación de dispositivos de baja potencia se encuentran listadas en la Norma Técnica ENACOM Q2-60.14[8]. Por otro lado, se deben tener en cuenta además los requisitos de ensayos para equipos de banda ancha para uso privado[15]. Si bien la homologación del desarrollo de este proyecto no es parte del alcance ni de los objetivos del mismo, la solución debe ajustarse a estas normas para usar eficiente de espectro electromagnético y seguir las normas legales vigentes de nuestro país.

Resumiendo las características técnicas que debe seguir un dispositivo de baja potencia que opera en la banda de frecuencias comprendida entre los 902–928MHz, se obtiene lo siguiente:

- En cada comunicación se efectuará la transmisión y recepción en la misma banda de frecuencias, utilizando la técnica de multiplexado en el tiempo, sincrónica o asincrónica (Protocolo de medición de Banda Ancha[15]).
- La anchura de banda medida entre los puntos del espectro de emisión que se encuentren 6 dB por debajo de la referencia tomada en la frecuencia de la portadora, no podrá ser menor que 500 kHz (Protocolo de medición de Banda Ancha[15]).
- Máxima potencia conducida en la banda de operación: no excederá de 1 W (Protocolo de medición de Banda Ancha[15]).
- La ganancia de antena podrá tomar valores tales que mantengan una potencia isotrópica radiada equivalente ("P.I.R.E.") de 4 W (6 dB W) como máximo (Protocolo de medición de Banda Ancha[15]).
- Los transmisores de baja potencia deben estar provistos de antenas integradas (permanentemente unidas al equipo) o de antenas específicas desmontables provistas de un conector especial. Dicho conector es aquel que no es del tipo normalizado que se encuentra disponible comercialmente o que no se utiliza normalmente para la conexión de RF. Así, es posible sustituir la antena externa en caso de avería solo por otra de iguales especificaciones (Norma Técnica ENACOM-Q2-60.14[8]).

- El nivel de intensidad de campo eléctrico, medido a una distancia de 3 m, estará limitado en cada banda a 50 000 µV m^{-1} (Norma Técnica ENACOM-Q2-60.14[8]).
- Las emisiones medidas fuera de la banda comprendida entre 902–928MHz, no superarán el nivel de 200 µV m^{-1} a 3 m medidos en un EPZA (Emplazamiento de Prueba de Zona Abierta) empleando un detector cuasi pico (Norma Técnica ENACOM-Q2-60.14[8]).
- Las emisiones no deseadas ubicadas por encima de 960 MHz estarán limitadas a 500 µV m^{-1} a 3 m medidos en un EPZA empleando un detector promedio. En caso de emplear un detector pico se considerará un límite 20 dB mayor (Norma Técnica ENACOM-Q2-60.14[8]).

En conclusión, para que un dispositivo que utilice el protocolo elegido en la banda de frecuencias no licenciada dispuesta por la UIT pueda operar legalmente dentro de las regulaciones de nuestro país, se deben respetar las limitaciones listadas anteriormente. Por lo que el proyecto es factible en términos legales si el mismo se ajusta a lo expuesto.

4.3. Factibilidad financiera

Teniendo en cuenta que el proyecto no contempla específicamente la implementación final en un dispositivo físico real y simplemente finalizar en la etapa de simulación, no es necesario más que una computadora para realizar la programación y las simulaciones correspondientes. Por otro lado el objetivo del trabajo es meramente académico y no se busca obtener un prototipo comercial del desarrollo. Es por estos motivos que no se consideró una evaluación de costos ni beneficios que limite al proyecto.

4.4. Alcance del proyecto

Después de analizar cuidadosamente el protocolo y todos los desafíos del proyecto se definió el alcance que tendría el mismo de forma tal de cumplir con los objetivos y redactar el informe correspondiente.

Para que el resultado final tenga un alcance claro y definido se propuso realizar la mayor parte de la capa física del protocolo seleccionado sin llegar a una implementación real. Esto incluiría una conexión con la capa superior para intercambiar datos hasta la señal en frecuencia intermedia de un paquete conformado como indica la especificación. De esta forma lo único que restaría de implementar sería la fase de radiofrecuencia (RF), es decir, un mezclador que lleve la señal desde la frecuencia intermedia hasta la señal pasabanda lista para ser filtrada y amplificada de forma analógica para luego ser transmitida por la antena.

Por otra parte, la especificación completa de la capa física fue tenida en cuenta, salvo las diferentes tasas de transmisión de datos. Por simplicidad, sólo se desarrolló la tasa de transmisión normal (Normal-Rate - Tabla 3.2). Sin embargo, la mayor parte de los bloques fueron desarrollados para satisfacer el caso más restrictivo, tratándose de la tasa de transmisión alta (Hi-Rate - Tabla 3.2), en lo que se refiere a latencia e intervalo mínimo de adquisición de datos válidos en la entrada. El transmisor desarrollado se diseñó para ser fácilmente escalable por lo que modificarlo para adoptar todas las tasas no debiera ser un problema desafiante.

Capítulo 5

Etapa de transmisión

En el siguiente capítulo se analiza el modelo conceptual de la cadena completa de transmisión y la implementación de cada bloque que la conforma. En primer lugar se discute el conjunto de elementos y a continuación se detalla cada uno de los bloques por separado con mayor detenimiento.

5.1. Modelo conceptual completo

En la Figura 5.1 se observa el esquema conceptual de la etapa de transmisión diseñado con las especificaciones de D7A descriptas anteriormente.

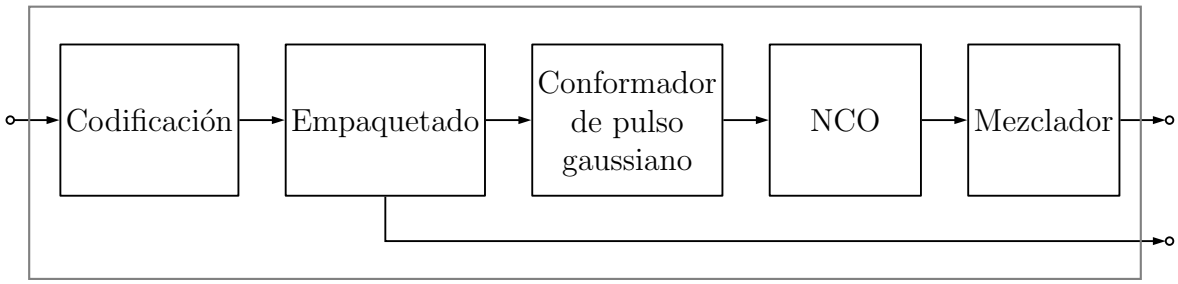


Figura 5.1: Esquema conceptual de la cadena de transmisión completa diseñada.

La entrada del sistema completo son los datos en forma de bytes provenientes de la capa superior junto con parámetros de configuración. Estos parámetros son el esquema de modulación (Tabla 3.2), la clase de codificación (Tabla 3.4) y la clase de palabra de sincronismo (Tabla 3.6).

La salida del sistema consiste en la señal ya modulada en frecuencia intermedia. Esta señal debería ser la entrada de la etapa de radiofrecuencia (RF) que se encarga de llevar la señal desde frecuencia intermedia a la frecuencia de portadora. Se eligió como frecuencia intermedia a $f_{if} = 12{,}5\,\text{MHz}$. La elección de esta frecuencia se justifica más adelante.

El bloque de Codificación consiste en un módulo cuya entrada son los bytes de datos y la clase de codificación especificada por la capa superior. A partir de esto, los datos son codificados de la forma que se detalla en la especificación del protocolo. La salida del bloque son los datos codificados en forma de bytes.

El bloque de Empaquetado consiste en un módulo cuya entrada son los bytes de datos ya codificados, el esquema de modulación (Tabla 3.2), la clase de codificación (Tabla 3.4) y la clase de palabra de sincronismo (Tabla 3.6). El objetivo de este bloque consiste en transformar los bits ingresados en símbolos con el formato de paquete especificado por el protocolo (Figura 3.3). El esquema de modulación indica la tasa de símbolos, y por ende el tiempo de símbolo (Tabla 3.2). La clase de codificación y la clase de palabra de sincronismo indican el valor de la palabra de sincronismo que se utiliza en el paquete (Tabla 3.6). El bloque posee dos salidas. Por un lado, un flujo de símbolos binarios en forma de escalón dependiendo del bit a transmitir. La segunda salida consiste en un valor que indica la ganancia o potencia del paquete y va directamente a la etapa de RF. Al principio toma valores incrementales, luego se mantiene en el valor máximo y al final del paquete baja a cero. El objetivo de esta salida es indicarle a la parte de RF la potencia de salida del paquete en cada momento. Es de esta forma que se obtienen las rampas ascendente y descendente que se especifican en la Figura 3.3 al principio y al final de cada paquete.

El bloque de Conformación de pulso gaussiano consiste en un módulo cuya entrada es el esquema de modulación (Tabla 3.2) y el flujo de símbolos en forma de escalón proveniente del empaquetado. El objetivo de este bloque es transformar los escalones que representan un símbolo en pulsos con forma gaussiana. Este componente suaviza las transiciones entre frecuencias dadas por cada símbolo de la modulación utilizada. Las ventajas y desventajas de este módulo se discuten más adelante.

El NCO u oscilador controlado numéricamente es un bloque que toma un valor de entrada y tiene como salida una señal sinusoidal de amplitud constante y frecuencia directamente proporcional al valor de la entrada (dentro de un rango de operación determinado). Es necesario aclarar que la salida en sentido estricto son dos señales sinusoidales que representan la componente en fase (Inphase o I) y la componente en cuadratura (Quadrature o Q) de un fasor que oscila a una frecuencia controlada.

Por último, el bloque Mezclador es un módulo que toma la señal y la multiplica por una sinusoidal de frecuencia fija para llevar la señal original a frecuencia intermedia f_{if}.

Es preciso aclarar que todo el transmisor podría haber sido implementado como un solo bloque usando HLS. Sin embargo, se decidió no hacerlo de forma monolítica ya que se pretendía ver el valor de los datos intermedios que viajan de módulo en módulo y poder depurar el proceso más fácilmente. Esta perspectiva agrega cierta modularidad que provee de claridad al modelo. Si se quisiera llevar el trabajo a un entorno no

académico, el primer paso para reducir los recursos que ocupa el transmisor sería hacerlo de forma monolítica, ya que se utiliza una considerable cantidad de recursos para establecer las interfaces entre ellos.

Las interfaces mencionadas (excepto aquellas que son parámetros de configuración o banderas) fueron implementadas utilizando el protocolo AXI4-Stream[16] para que las mismas sean estándar y el bloque sea reutilizable en otro lugar sin necesidad de modificaciones. Esta decisión fue tomada debido a la simplicidad de indicarle a HLS el estándar de la interfaz. En adelante, este tipo de interfaces serán referenciadas como interfaces AXI4-Stream[16].

Para el diseño del sistema se utilizó una frecuencia de reloj $f_{clock} = 100\,\text{MHz}$. Todos los bloques utilizan la misma frecuencia.

5.2. Bloque de codificación

El diseño posee tres entradas. Estas son los datos en forma de bytes, el esquema de codificación representado por 2 bit (índice de la Tabla 3.4) y una señal de inicio representada por 1 bit que se utiliza como bandera para iniciar el proceso de codificado, como se muestra en la Tabla 5.1. La entrada de datos se implementó con interfaz AXI4-Stream[16] como una cola o FIFO (First In-First Out) para desacoplar el flujo de datos que proviene de la capa superior con el bloque de codificación. Esto también permite disminuir la atención que se pone en la latencia del módulo y del intervalo de tiempo mínimo entre dos bytes de entrada. Asimismo, la salida, que también es en forma de bytes, se implementó con otra cola y también con interfaz AXI4-Stream[16]. Una vez que la bandera se levanta, el bloque considera que la FIFO de entrada ya está completa e inicia el proceso.

Señal (Puerto)	I/O	Tipo	Descripción
input_data	Entrada	`Bus AXI4-Stream`	Datos binarios (bytes) a transmitir.
coding_scheme	Entrada	`STD_LOGIC_VECTOR[1:0]`	Esquema de codificación (Tabla 3.4).
start_coding	Entrada	`STD_LOGIC_VECTOR[0:0]`[1]	Bandera de comienzo de transmisión.
output_data	Salida	`Bus AXI4-Stream`	Datos binarios (bytes) codificados.

1 Tipo de interfaz establecido por HLS para entradas de 1 bit.

Tabla 5.1: Interfaces del bloque de codificación.

Los datos de entrada siguen el flujo de codificación de la forma que se indica en el diagrama de flujos de la Figura 5.2. Una vez que los datos ingresan al módulo pueden tomar dos caminos como se puede ver en la figura.

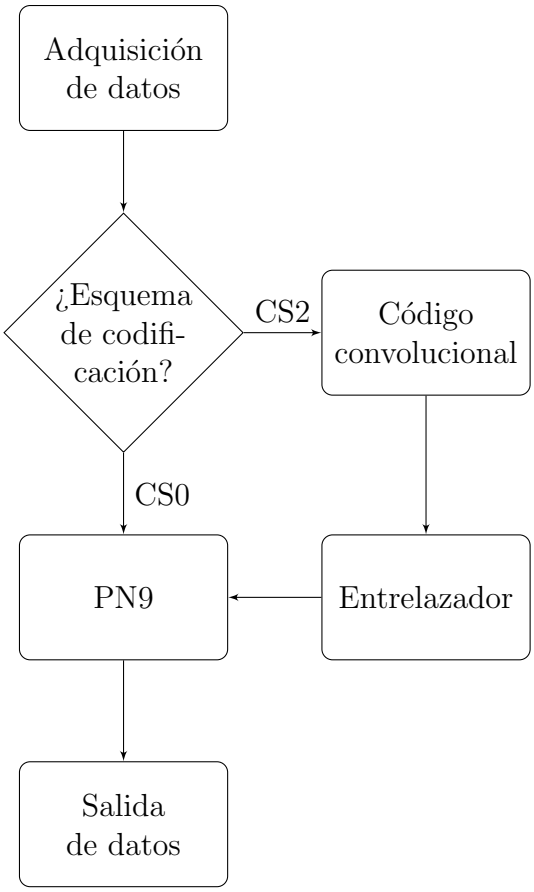


Figura 5.2: Diagrama de flujo del proceso de codificación. CS0 y CS2 son esquemas de codificación especificados en la Tabla 3.4.

Resultados de pruebas

Para verificar el funcionamiento de cada parte del codificador, se eligió implementar primero todas las etapas por separado y corroborar que el funcionamiento de cada una de ellas sea el correcto. Para esto se realizó un cuidadoso seguimiento de las variables intermedias en la fase de pruebas funcionales de alto nivel (primera etapa de HLS) y se pudo confirmar que el proceso era correcto. Las salidas fueron validadas utilizando como referencia una una tabla realizada manualmente.

Dado que cada parte del codificador no tiene mucha complejidad por si misma, como prueba inicial esto fue suficiente.

Una vez que todas las etapas fueron unidas, se realizó la codificación de un conjunto de datos por ambos esquemas. Más adelante, una vez implementado el decodificador, se verificó que el camino completo funcionara de forma adecuada.

5.3. Bloque de empaquetado

Como se mencionó anteriormente, el módulo tiene dos objetivos principales. El primero consiste en tomar los datos codificados y ensamblarlos en un paquete con la forma que se muestra en la Figura 3.3 dada por la especificación del protocolo. El segundo reside en transformar los datos binarios en muestras con forma de pulsos cuadrados.

Para lograr lo expuesto, se diseñó una máquina de estados para gestionar de forma ordenada la estructura del paquete y las muestras de salida. Este diseño no era estrictamente necesario utilizando HLS, pero aportó mucha claridad a la estructura del código del programa. La forma de implementarlo sin una máquina de estados es desacoplando los dos objetivos del módulo en bloques separados. De igual forma, la síntesis de HLS termina implementando el modelo como una máquina de estados, por lo que no se encontró motivos para no explicitar los mismos en el código. El esquema de dicha máquina de estados se muestra en la Figura 5.3.

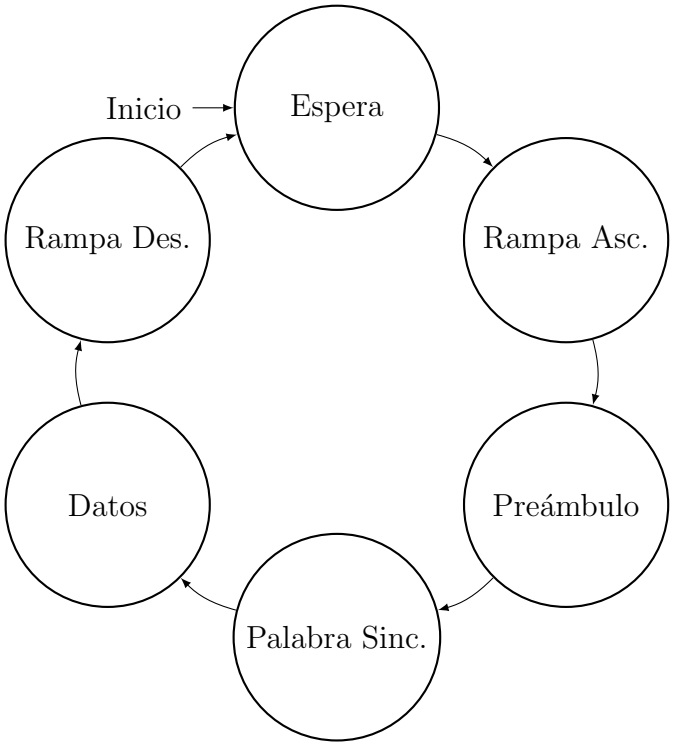


Figura 5.3: Máquina de estados utilizada para ordenar la estructura del paquete y las muestras de salida.

El bloque posee cuatro entradas. Estas consisten en los datos ya codificados en forma de bytes, el esquema de modulación en forma de 2 bit (índice de la Tabla 3.2), el esquema de codificación también en forma de 2 bit (índice de la Tabla 3.4) y la clase de palabra de sincronismo en forma de 1 bit (Tabla 3.6). La entrada de datos se implementó con interfaz AXI4-Stream[16] como una cola o FIFO (First In-First Out) para desacoplar el flujo de datos que proviene del módulo anterior con el bloque de

empaquetado. Esto también permite disminuir la atención que se pone en la latencia del módulo y del intervalo mínimo entre dos bytes de entrada. Todas las interfaces del bloque se muestran en la Tabla 5.2.

Se implementaron dos salidas. Por un lado las muestras de cada símbolo, en forma de 2 bit. Por otra parte, un valor de ganancia representado con 8 bit(sin signo). Las muestras se encaminan al siguiente módulo, es decir al conformador de pulso gaussiano, y representan los símbolos a enviar. Es el primer paso para generar la señal analógica en función de los datos binarios. La otra salida es una ganancia que se dirige directamente a la etapa de RF, es decir es una salida del sistema completo. Esta ganancia es la que da como resultado la rampa ascendente y descendente al principio y final del paquete respectivamente. Además durante el resto de las etapas del paquete se mantiene en su valor máximo para indicar al transmisor que no modifique la amplitud de la señal.

Señal (Puerto)	**I/O**	**Tipo**	**Descripción**
input_data	Entrada	`Bus AXI4-Stream`	Datos binarios (bytes) codificados.
coding_scheme	Entrada	`STD_LOGIC_VECTOR[1:0]`	Esquema de codificación (Tabla 3.4).
modulation_scheme	Entrada	`STD_LOGIC_VECTOR[1:0]`	Esquema de modulación (Tabla 3.2).
sync_word_class	Entrada	`STD_LOGIC_VECTOR[0:0]`[1]	Clase de palabra de sincronismo (Tabla 3.6).
output_data	Salida	`Bus AXI4-Stream`	Muestras del símbolo: `01` para el símb. `1` `11` para el símb. `0`
ramp_out	Salida	`STD_LOGIC_VECTOR[7:0]`	Valor de ganancia en etapa RF (sin signo).

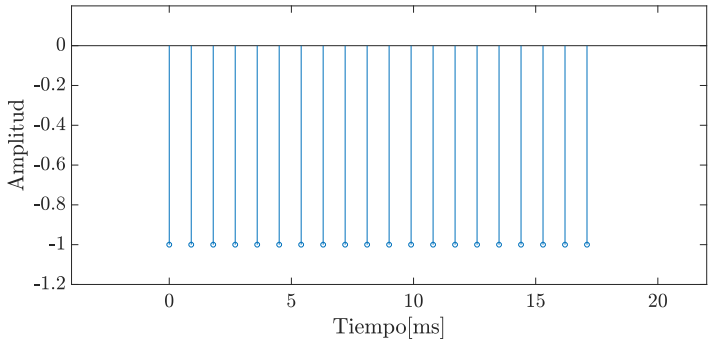
1 Tipo de interfaz establecido por HLS para entradas de 1 bit.

Tabla 5.2: Interfaces del bloque de empaquetado.

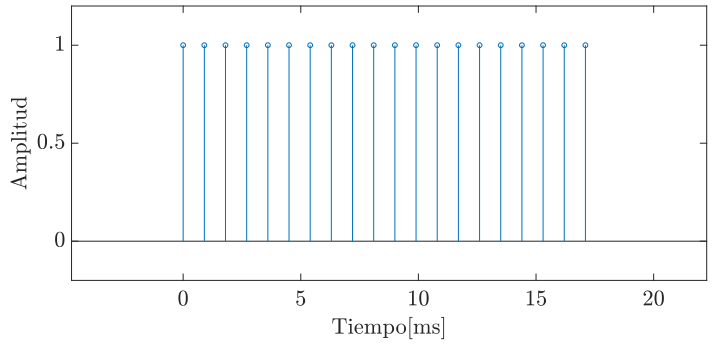
Para determinar la cantidad de símbolos que se utilizaría en cada etapa del paquete se utilizaron los valores típicos que se muestran en la Tabla 3.5. Si bien esto fue implementado de forma fija, puede cambiarse muy fácilmente en el código para que la cantidad de símbolos de cualquier parte del paquete dure lo que se necesite.

Se eligió arbitrariamente que cada pulso tenga 20 muestras. Es decir que para una tasa normal de transmisión (Normal-Rate - Tabla 3.2) en donde el tiempo de símbolo es $T_s = 18\,\mu s$, el tiempo de muestra sería igual a $T_m = T_s/20 = 0{,}9\,\mu s$. Este tiempo de

muestra permite un buen margen de tiempo para ejecutar las operaciones necesarias entre cada muestra y además provee una buena resolución temporal del pulso. Aún para el caso más complicado como es la tasa de transmisión alta (Hi-Rate - Tabla 3.2), en donde el tiempo de símbolo es $T_s = 6\,\mu s$, el tiempo de muestra sería igual a $T_m = T_s/20 = 0{,}3\,\mu s$ y la justificación anterior también vale. En la Figura 5.4 se puede ver la representación temporal de los pulsos de salida del símbolo 0 (Figura 5.4a) y del símbolo 1 (Figura 5.4b).



(a) Representación temporal del pulso cuadrado usada para el símbolo 0.



(b) Representación temporal del pulso cuadrado usada para el símbolo 1.

Figura 5.4: Representación temporal de las muestras que conforman los pulsos cuadrados utilizados para ambos símbolos en el bloque de empaquetado.

Resultados de pruebas

Para validar el modelo implementado, se realizó la co-simulación de la implementación y se verificó la forma de onda resultante. Para ello se ingresaron ocho bytes arbitrarios al módulo y se observó la evolución en el tiempo de las muestras para corroborar la forma de la onda saliente. Los bytes ingresados en formato hexadecimal fueron "`0xF0 0xF0 0xF0 0xF0 0x00 0x00 0x00 0xFF`". Se eligió estos datos de entrada por el simple motivo de poder diferenciar fácilmente las partes del paquete a lo largo del tiempo. En la Figura 5.5 se puede observar la salida que se obtuvo del módulo en la Co-simulación del mismo.

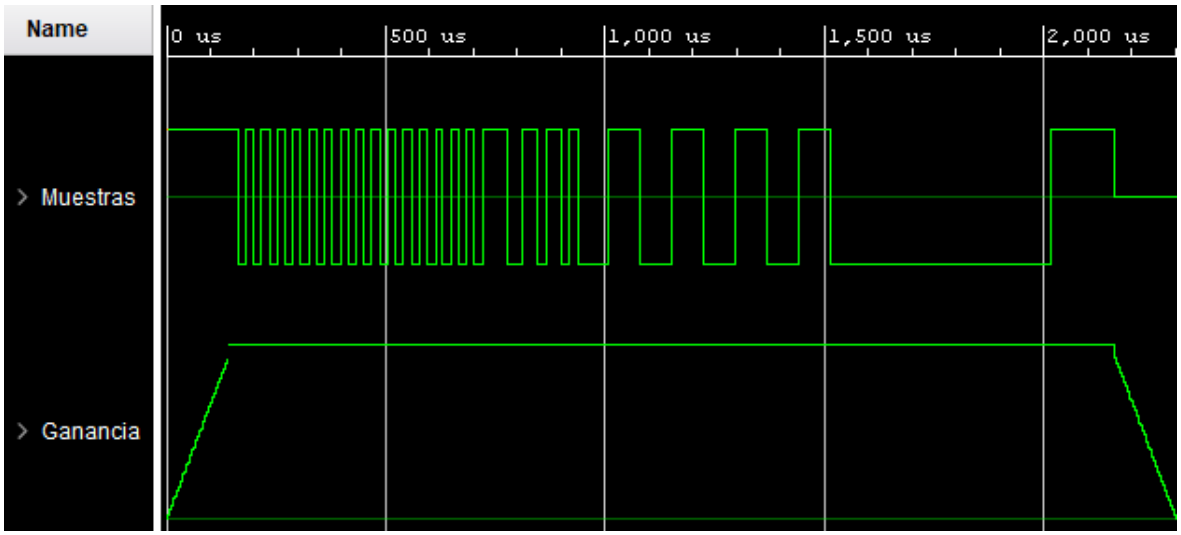


Figura 5.5: Forma de onda resultante de la co-simulación del módulo de empaquetado de D7A. La forma de onda superior es la salida de las muestras, el paquete ya formado representado por pulsos cuadrados. El valor mínimo que toma esta señal es -1 y el valor máximo es 1. La forma de onda inferior es la ganancia de potencia que sale directamente hacia la etapa de RF. El valor mínimo que toma esta señal es 0, y el valor máximo es 255.

Se puede observar en la Figura 5.5 ambas salidas del módulo. En la parte superior está el paquete propiamente dicho representado con pulsos cuadrados como se describió anteriormente. En la parte inferior se puede ver la ganancia en potencia que va directamente a la etapa de RF. En esta salida se puede ver las rampas de potencia ascendente y descendente al comienzo y final del paquete respectivamente como se indicó en las especificaciones (Figura 3.3).

Se puede diferenciar fácilmente las partes del paquete en la formación de pulsos de la onda superior de la Figura 5.5. En la primer parte consta el preámbulo que consiste en una secuencia alternada de unos y ceros durante 32 símbolos (desde los 144 µs hasta los 720 µs). Luego le sigue la etapa de la palabra de sincronismo durante 16 símbolos (desde los 720 µs hasta los 1008 µs). Después le sigue la cadena de datos que se ingreso, que en formato hexadecimal sería "`0xF0 0xF0 0xF0 0xF0 0x00 0x00 0x00 0xFF`" (desde los 1008 µs hasta los 2160 µs). Además se verificó que cada símbolo tuviera las 20 muestras esperadas y que las mismas estuvieran distanciadas 0,9 µs. Por otra parte, la especificación no determina cómo tiene que ser la señal durante las rampas ascendente y descendente. Es por esto que se decidió que la salida de las muestras durante la rampa ascendente se correspondan al símbolo 1 ya que siempre es el primer símbolo del preámbulo, y durante la rampa descendente sólo sea transmitida la portadora. Siendo que el bloque tuvo la salida esperada, se consideró que la implementación era válida.

Cuando ambas señales salen del módulo están sincronizadas. Sin embargo, a medida que una de ellas viaja y sigue siendo procesada por el resto de los módulos, cuando llegan al final del transmisor, existe un desfasaje entre las dos señales de salida dado

por la latencia que aportan el resto de los módulos a las muestras de los símbolos. A pesar de ello, este desfasaje representa alrededor de un 2 % del tiempo de símbolo para una tasa de transmisión normal (Normal-Rate - Tabla 3.2), alrededor de un 6 % del tiempo de bit para una tasa de transmisión alta (Hi-Rate - Tabla 3.2) y menos del 1 % del tiempo de bit para una tasa de transmisión baja (Low-Rate - Tabla 3.2). Es por esto que no se consideró un problema significativo. De todas formas solucionarlo sería simplemente agregar un módulo que le sume una latencia arbitraria a la salida de la ganancia que sale del transmisor. Ambas salidas se implementaron con interfaz AXI4-Stream[16]. La salida de muestras se ingresa a una pequeña cola para que la comunicación con el siguiente módulo no sea bloqueante y se mantenga el temporizado de las muestras si algo llega a fallar en los módulos siguientes.

5.4. Bloque de Conformación de pulso gaussiano

La especificación del protocolo detalla que para todos los esquemas de modulación se debe usar 2-(G)FSK (Tabla 3.2). Esto indica que antes de modular la señal utilizando FSK, se puede ingresar a un filtro gaussiano para que los símbolos tengan forma gaussiana y las transiciones entre ellos sean más suaves. Según la especificación podría utilizarse la modulación FSK simple, pero el filtro gaussiano posee sus ventajas. Este tipo de implementación ayuda a limpiar espectralmente la señal debido a que atenúa las frecuencias que están fuera de las bandas de interés. En contraparte, los símbolos sufren una degradación debido a las "colas" de los símbolos aledaños, por lo que su amplitud varía y genera interferencia intersímbolo o ISI (por sus siglas en inglés). Dado que el protocolo está diseñado para escenarios en los que existen muchos dispositivos conviviendo, es deseable que exista la menor interferencia posible entre ellos.

La forma de parametrizar estos filtros es a través del producto ancho de banda-tiempo de bit o también referenciado como WT_b[17], donde el ancho de banda W es el ancho de banda en banda base de 3 dB del filtro conformador. A medida que este producto se acerca a cero, la gaussiana se dispersa más en el tiempo y los efectos mencionados anteriormente se intensifican. Por el contrario, a medida que el producto aumenta, más se acerca al modelo sin filtro gaussiano.

Se evaluaron algunos coeficientes teniendo en cuenta esta relación de compromiso a fin de seleccionar uno para la implementación. Los coeficientes evaluados fueron $WT_b = 0{,}3$, $WT_b = 0{,}5$, $WT_b = 0{,}7$ y $WT_b = 0{,}9$. Se analizaron estos coeficientes debido a que el protocolo impone como valores posibles los que están en el rango entre 0,3 y 1. En primer lugar se analizó la atenuación de lóbulos laterales en el espectro que produce cada uno de estos coeficientes. El resultado obtenido se muestra en la Figura 5.6.

Se puede ver en la Figura 5.6 que todos los coeficientes tienen una atenuación

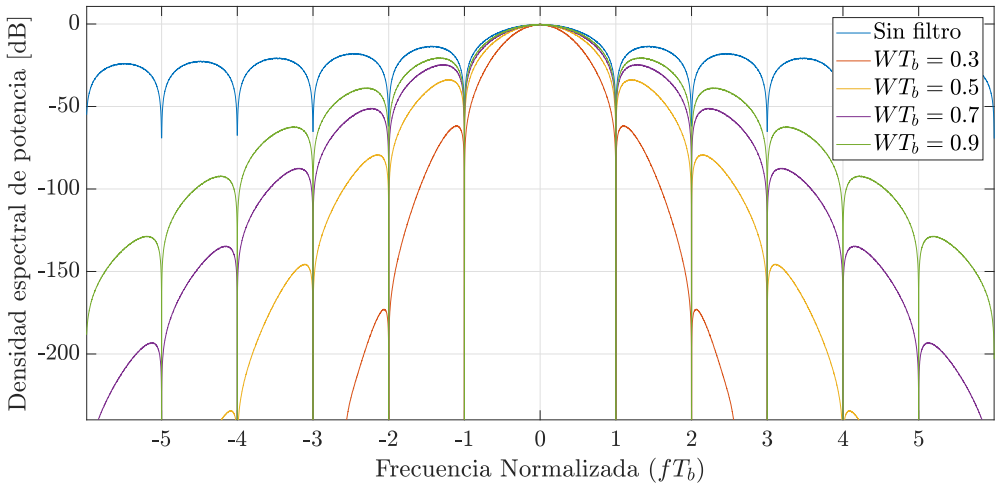


Figura 5.6: Comparativa de la Densidad espectral de potencia de una señal procesada por diversos conformadores de pulsos gaussianos con diferentes productos ancho de banda-tiempo de bit WT_b (antes de ser modulado por el NCO).

considerable de los lóbulos laterales del espectro de la señal con forma de seno cardinal. Sin embargo existe bastante diferencia entre los diferentes productos. En este aspecto el producto $WT_b = 0{,}3$ es mejor que las otras opciones, siguiéndole $WT_b = 0{,}5$, $WT_b = 0{,}7$ y por último $WT_b = 0{,}9$. Se analizaron los diferentes valores de atenuación en el máximo local del primer lóbulo secundario y se obtuvieron los valores de la Tabla 5.3.

Producto WT_b	**Atenuación**
0.3	75,3 dB
0.5	27,1 dB
0.7	13,8 dB
0.9	8,4 dB

Tabla 5.3: Atenuación del máximo local del primer lóbulo secundario utilizando conformadores de pulso gaussianos con diferentes productos WT_b.

Se realizó un análisis adicional simulando la modulación de un mensaje aleatorio utilizando los filtros mencionados y verificar la atenuación que provee cada uno de ellos fuera del canal. En la Figura 5.7 se puede ver el resultado obtenido de la simulación.

Se puede ver en la Figura 5.7 que la atenuación del espectro en los canales contiguos es notable, por lo que la utilización de un conformador gaussiano provechosa en canales donde hay muchos dispositivos conviviendo. Se observa además que a medida que el producto WT_b es menor, la contaminación que produce en otros canales es menor.

Para evaluar la interferencia intersímbolo o ISI se examinó el diagrama de ojos resultante de cada uno de los conformadores de pulso analizados. Los diagramas de ojo realizados se pueden ver en la Figura 5.8.

Utilizando el criterio de la distorsión pico, se puede ver en la Figura 5.8 que la

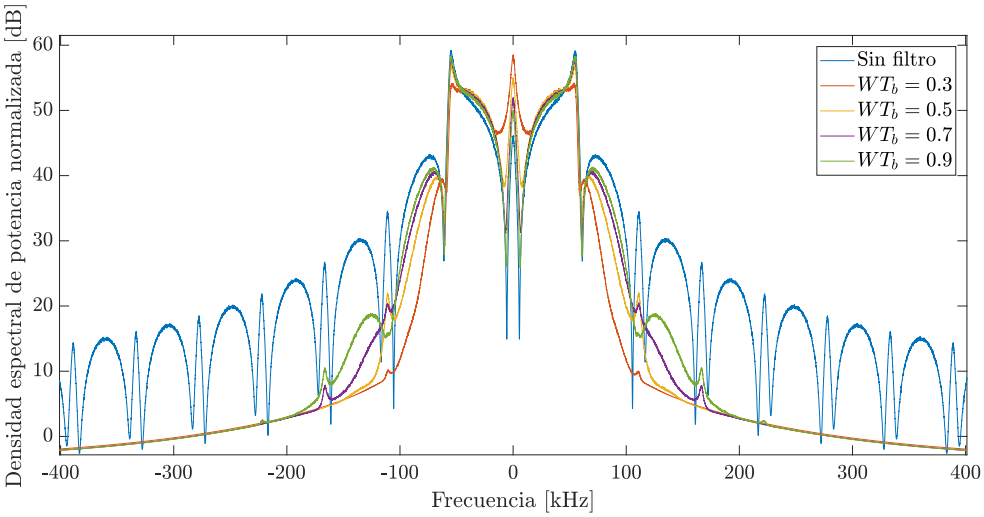


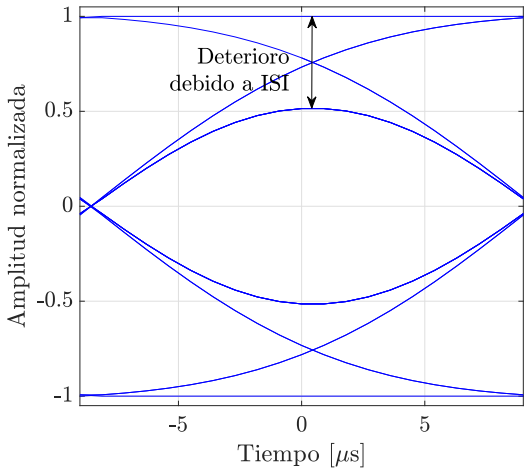
Figura 5.7: Comparativa de la Densidad espectral de potencia de una señal procesada por diversos conformadores de pulsos gaussianos con diferentes productos ancho de banda-tiempo de bit WT_b luego de ser modulada para una tasa de transmisión normal (Normal-Rate). El espaciamiento entre canales para esta tasa de transmisión es de 200 kHz según la especificación del protocolo (Tabla 3.2).

degradación que produce cada filtro es muy variada. Como se esperaba, al contrario que en el análisis anterior, en este aspecto el producto $WT_b = 0{,}9$ es mejor que las otras opciones, siguiéndole $WT_b = 0{,}7$, $WT_b = 0{,}5$ y por último $WT_b = 0{,}3$. Se analizaron los diferentes valores de degradación máxima y se obtuvieron los valores de la Tabla 5.4.

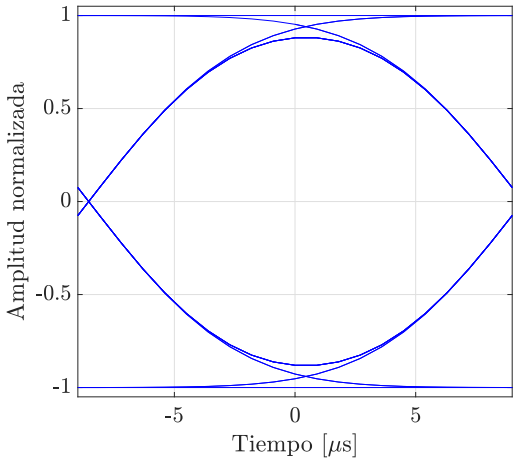
Producto WT_b	**Degradación**
0.3	≈5,8 dB
0.5	≈1 dB
0.7	≤0,2 dB
0.9	≤0,02 dB

Tabla 5.4: Degradación observada en la señal debido a la interferencia intersímbolo o ISI utilizando conformadores de pulso gaussianos con diferentes productos WT_b.

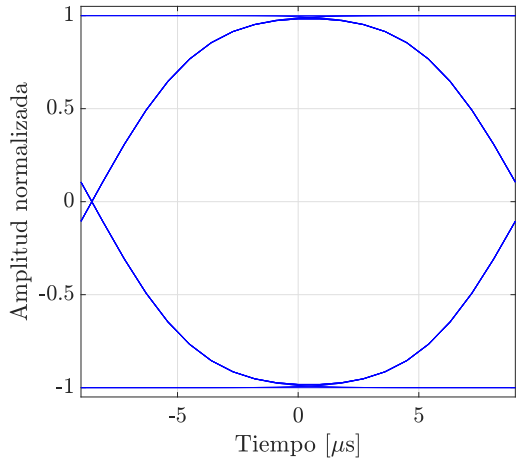
Habiendo evaluado ambos efectos en la elección del filtro conformador de pulso gaussiano, se prosiguió a tomar una decisión. Por un lado el filtro con producto $WT_b = 0{,}9$ presenta una mejora en el espectro bastante inferior al resto más allá de poca degradación de la señal por ISI. En oposición, el filtro con producto $WT_b = 0{,}3$ es el caso opuesto, siendo superior al resto comparando la atenuación de los lóbulos laterales, pero con una degradación de señal muy significativa. Por lo visto, ambos candidatos son descartados. Los casos más equilibrados son los filtros con producto $WT_b = 0{,}5$ y $WT_b = 0{,}7$. En un escenario en donde se espera que hayan muchos dispositivos operando en la misma banda de frecuencia en canales contiguos, y que además el protocolo tiene mecanismos de detección y corrección de errores no sólo en la capa física, se considera



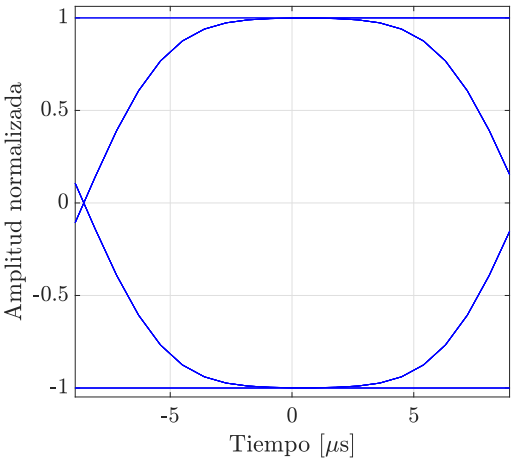
(a) Diagrama de ojo del filtro gaussiano con $WT_b = 0{,}3$



(b) Diagrama de ojo del filtro gaussiano con $WT_b = 0{,}5$



(c) Diagrama de ojo del filtro gaussiano con $WT_b = 0{,}7$



(d) Diagrama de ojo del filtro gaussiano con $WT_b = 0{,}9$

Figura 5.8: Diagramas de ojo resultante de cada uno de los conformadores de pulso gaussianos con diferentes productos ancho de banda-tiempo de bit WT_b.

más importante el hecho de tener una contaminación espectral menor en los canales aledaños. Es por este motivo que se eligió el filtro con producto $WT_b = 0{,}5$ para la implementación del conformador de pulso gaussiano.

En condiciones ideales la convolución de la señal con el filtro gaussiano es de duración infinita, pero esto resulta imposible de realizar. Es por esto que el largo de la forma de onda gaussiana debe ser limitada. Para tomar una decisión de donde truncar la convolución mencionada, se evaluaron varios filtros con producto $WT_b = 0{,}5$ pero con ventanas de tiempo cuadradas de duración distinta. Se analizaron los casos en que la ventana toma $1T_b$, $2T_b$, $3T_b$ y $4T_b$. Los resultados obtenidos se muestran en la Figura 5.9.

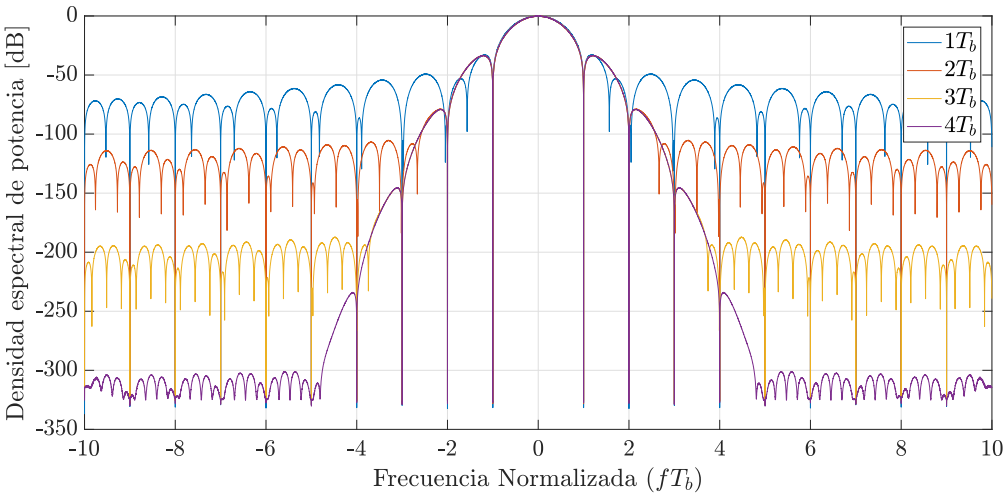


Figura 5.9: Densidad espectral de potencia obtenida al convolucionar la señal con filtros gaussianos con un producto $WT_b = 0{,}5$ y diferentes ventanas temporales (antes de ser modulado por el NCO).

Se puede ver que el efecto de recortar la ventana temporal de la gaussiana es ponerle un piso a la atenuación dada por el filtro gaussiano de los lóbulos laterales. A menor soporte temporal, menor es la atenuación lograda. Más allá que entre todas las opciones hay bastante diferencia en la atenuación lograda, todas son bastante buenas. Por otra parte, tener una ventana muy grande significa operar muchas veces antes de generar una salida, incrementar la latencia y además tener una representación numérica suficiente, debido a que las colas de la gaussiana poseen valores muy pequeños y se requiere un elevado rango dinámico en las operaciones. Es por esto que se eligió una ventana temporal de $2T_b$, ya que posee una buena relación de compromiso entre la mejora lograda y el costo de su implementación.

Una vez implementado se observó que los valores extremos de la función gaussiana (es decir los más pequeños) tenían valores de cienmilésimas. En particular se utilizaron 16 bit para representar la parte decimal (utilizando punto fijo), por lo que estas colas eran representadas con los mínimos valores posibles. Esto implica que agregar una

ventana (función), donde generalmente estas colas son atenuadas por algún factor, no tendría representación posible con la cantidad de bits utilizados. A priori se consideró que la cantidad de recursos que se requieren para operar con representaciones más grandes no justificaban el beneficio de utilizar una ventana ya que el piso de atenuación es bajo (Figura 5.9). La evaluación de tal relación de compromiso no se incluyó en el alcance del proyecto y se deja de lado como una optimización futura.

El bloque implementado posee simplemente una entrada y una salida. La entrada consiste en las muestras provenientes del bloque empaquetador representadas con 2 bit. La salida consiste en las muestras del pulso ya conformado por el filtro gaussiano representadas utilizando 25 bit (1 bit para el signo y el resto para el valor). Estas interfaces se observan en la Tabla 5.5.

Señal (Puerto)	I/O	Tipo	Descripción

signal_in	Entrada	`Bus AXI4-Stream`	Muestras del símbolo:
			`01` para el símb. `1`
			`11` para el símb. `0`

signal_out	Salida	`Bus AXI4-Stream`	Muestras del pulso conformado (25 bit con signo).

Tabla 5.5: Interfaces del bloque de conformación de pulso gaussiano.

Resultados de pruebas

Para validar el modelo implementado, se realizó la co-simulación de la implementación y se verificó la forma de onda resultante. Para ello se ingresaron once símbolos arbitrarios representados por veinte muestras cada uno y se observó la evolución en el tiempo de la salida para corroborar la forma de la onda saliente. Los símbolos ingresados fueron "`0 0 1 1 0 1 1 1 1 1 0`". Es necesario aclarar que no fueron ingresados en el tiempo que dice la especificación, sólo se probó que el comportamiento sea el indicado. El temporizado entre muestra y muestra de salida depende directamente de la entrada, por lo que las mismas estarían distanciadas el mismo tiempo que la entrada manteniéndose el temporizado de las muestras. La validación de la duración del símbolo se realizó en las pruebas de integración. En la Figura 5.10 se puede observar la salida que se obtuvo del módulo en la co-simulación del mismo.

Se puede observar en la Figura 5.10 la salida del módulo. Se puede diferenciar claramente los símbolos representados por los pulsos positivos y negativos conformados por el filtro gaussiano. El valor máximo que alcanzan los pulsos tiene un valor en particular elegido en base al siguiente módulo, por lo que se explicará más adelante.

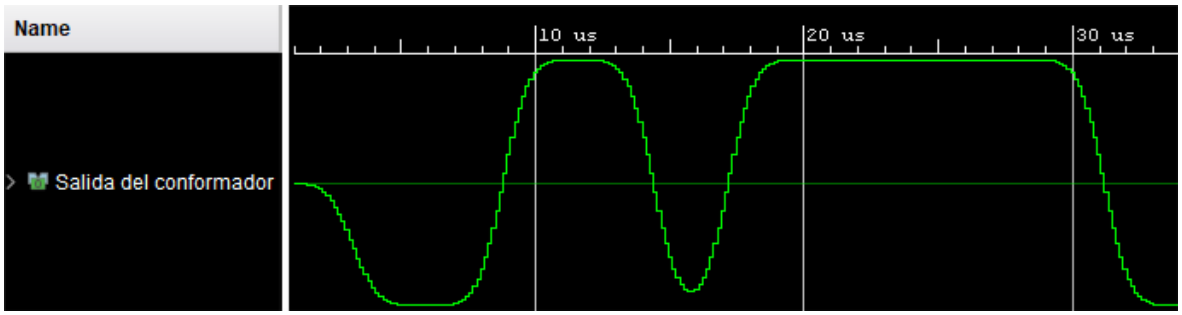


Figura 5.10: Forma de onda resultante de la co-simulación del módulo de conformación de pulsos gaussianos de D7A. La forma de onda representa la salida de las muestras, es decir, los símbolos representados por pulsos gaussianos. Los símbolos ingresados fueron "`0 0 1 1 0 1 1 1 1 1 0`". El valor mínimo que toma la señal es -16757 y el valor máximo es 16757. Estos valores se justifican más adelante.

5.5. Bloque NCO

El Oscilador Controlado Numéricamente o NCO (por sus siglas en inglés) es un módulo que genera señales digitales representando una forma de onda sinusoidal de tiempo discreto y amplitud constante. La frecuencia de la sinusoide de salida es directamente proporcional al valor de la entrada en un rango de operación determinado. La forma clásica de implementar un NCO es con un acumulador de fase y una tabla de búsqueda o LUT (por sus siglas en inglés) donde está representado la onda sinusoidal. La salida del acumulador de fase indica de que parte de la LUT sacar el dato de salida. A medida que el acumulador se incrementa, se va recorriendo esta LUT y se va generando la sinusoide a la salida. A mayores valores de entrada, el acumulador crece más rápido y la LUT es recorrida más rápidamente también, generando una sinusoidal de mayor frecuencia. Análogamente si la entrada disminuye.

Para implementar este módulo se utilizó una librería propia de HLS en donde la funcionalidad completa ya está desarrollada. Lo único que se requirió hacer es configurar ciertos parámetros y las interfaces del bloque. Los parámetros a configurar fueron el número de bits utilizados en el acumulador de fase, el número de bits utilizado para referenciar la LUT y por último el número de bits para representar la salida de la función. La cantidad de bits que define el acumulador de fase determina la precisión de la frecuencia que puede ser sintetizada. La cantidad de bits usada para referenciar la LUT define la resolución de la tabla (es decir, la cantidad de muestras de la onda sinusoidal). Por último, la cantidad de bits de la salida define la resolución en amplitud de la onda sinusoidal.

El bloque posee las entradas y salidas que requiere la función de la librería. Por un lado las entradas son el incremento del acumulador y la fase inicial de la señal u offset. El incremento del acumulador es una entrada de 32 bit que en realidad se utilizan los 25 bit menos significativos (esta adaptación se realiza debido a que las interfaces AXI4-Stream[16] tienen un tamaño de bits múltiplo de 8). Esta cantidad de bits utilizada

para la entrada va de la mano con la cantidad de bits para el acumulador de fase que se definió en el NCO. La cantidad de bits que se utiliza para el offset tiene que ser igual a la del incremento del acumulador. Por otra parte, las salidas del módulo son las componentes en fase (I) y en cuadratura (Q). Se eligió que estas fueran representadas con 16 bit, debido a que esta representación presenta un buena resolución de la onda senoidal saliente. Las interfaces del bloque se observan en la Tabla 5.6.

Señal (Puerto)	I/O	Tipo	Descripción
pinc	Entrada	`Bus AXI4-Stream`	Incremento del acumulador (25 bit con signo).
poff	Entrada	`Bus AXI4-Stream`	Fase inicial (25 bit con signo).
outputVal_real	Salida	`Bus AXI4-Stream`	Componente en fase (I) (16 bit con signo).
outputVal_imag	Salida	`Bus AXI4-Stream`	Componente en cuadratura (Q) (16 bit con signo).

Tabla 5.6: Interfaces del bloque NCO.

Dado que la salida tiene que ser una sinusoidal que posea las frecuencias de la Tabla 3.2, se requirió que el NCO tenga la resolución en frecuencia necesaria para que se cumpla este objetivo de forma aceptable. La resolución en frecuencia que tenga depende de la frecuencia de reloj del sistema y de la cantidad de bits con la que se representa el acumulador de fase (como se dijo anteriormente). Esta frecuencia mínima de resolución f_{res} está dada por la siguiente ecuación, donde N es la cantidad de bits con la que se representa el acumulador de fase y f_{clock} es la frecuencia de reloj de sistema:

$$f_{res} = \frac{f_{clock}}{2^N} \tag{5.1}$$

Dado que se utilizó una frecuencia de reloj $f_{clock} = 100\,\text{MHz}$, para alcanzar una resolución aceptable se tomo una cantidad $N = 25\,\text{bit}$. De esta forma se logró que la resolución fuera la siguiente:

$$f_{res} = \frac{100\,\text{MHz}}{2^{25}} \approx 2{,}98\,\text{Hz} \tag{5.2}$$

Para calcular el valor que tiene que tener la señal de entrada para generar la salida correspondiente A_{NR}, se realizó el siguiente cálculo, teniendo en cuenta la ecuación 5.1, donde $f_{NR} = \pm 50\,\text{kHz}$ es la frecuencia de salida correspondiente al símbolo 1 (signo

positivo) y símbolo 0 (signo negativo) según la Tabla 3.2 para una tasa de transmisión normal (Normal-Rate):

$$A_{NR} = \frac{f_{NR}}{f_{res}} = \frac{2^N f_{NR}}{f_{clock}} \approx \pm 16\,777 \tag{5.3}$$

Análogamente se realizaron los cálculos para el resto de las tasas de transmisión, aunque no hayan sido implementadas. Se obtuvieron los siguientes valores, donde A_{HR} y A_{LR} son los valores que tiene que tener la señal de entrada para generar la salida deseada para una tasa de transmisión alta (Hi-Rate) y para una transmisión baja (Low-Rate) respectivamente, $f_{HR} = \pm 41{,}667\,\text{kHz}$ es la frecuencia de salida correspondiente al símbolo 1 (signo positivo) y símbolo 0 (signo negativo) según la Tabla 3.2 para una tasa de transmisión alta (Hi-Rate) y $f_{LR} = \pm 4{,}800\,\text{kHz}$ es la frecuencia de salida correspondiente al símbolo 1 (signo positivo) y símbolo 0 (signo negativo) según la Tabla 3.2 para una tasa de transmisión baja (Low-Rate):

$$A_{HR} = \frac{f_{HR}}{f_{res}} = \frac{2^N f_{HR}}{f_{clock}} \approx \pm 13\,981 \tag{5.4}$$

$$A_{LR} = \frac{f_{LR}}{f_{res}} = \frac{2^N f_{LR}}{f_{clock}} \approx \pm 1611 \tag{5.5}$$

Estos valores son aproximados ya que se tienen que ser redondeados a un valor entero. En todos los casos el error obtenido por redondeo es menor al 0,025 %. Se podría haber reducido la resolución de forma tal de necesitar menos recursos para la síntesis, pero como se trata de un módulo relativamente poco demandante se prefirió no hacerlo.

Estos valores de amplitudes fueron con los que se configuraron el bloque anterior de conformación de pulso gaussiano. Es decir que indicándole al conformador el esquema de modulación, este configura la salida para que la el valor máximo del pulso se aproxime a A_{NR}, A_{HR} o A_{LR} según el caso.

Se especificó que tuviera interfaces AXI4-Stream[16] y que la síntesis adopte el esquema de tubería o pipeline para que el bloque pueda tomar valores de entrada con el mínimo intervalo de tiempo posible entre ellos. Ambas configuraciones se establecieron mediante el uso de pragmas.

Resultados de pruebas

Una vez que el módulo fue desarrollado, se probó con las entradas A_{NR}, A_{HR} y A_{LR} y se verificó que la salida senoidal tuviera el período esperado. En la Figura 5.11 se muestra la co-simulación obtenida para un valor de entrada dado por $A_{NR} = 16\,777$.

Se observa en la Figura 5.11 que el período de las formas de onda salientes es $T_{NR} \approx 20{,}08\,\mu\text{s}$, por lo que su frecuencia es muy parecida a la esperada de $f_{NR} = +50\,\text{kHz}$. Esto

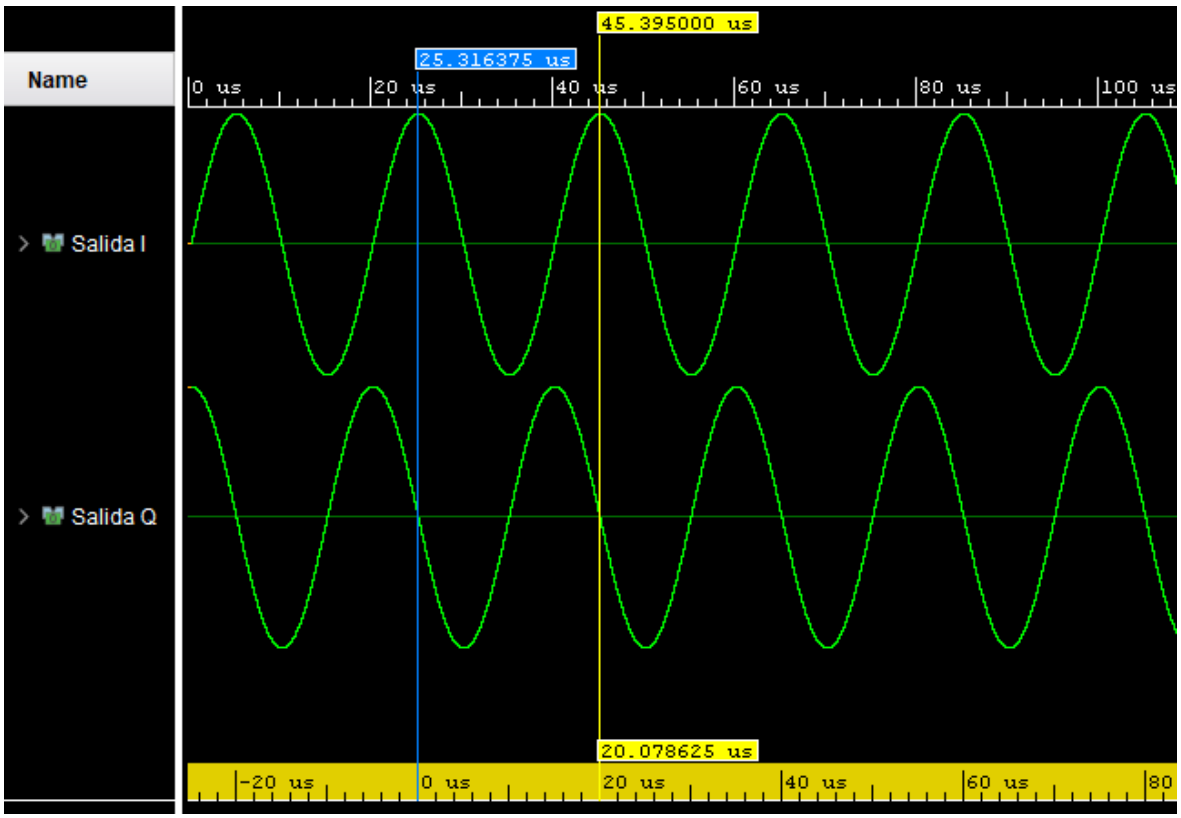


Figura 5.11: Forma de onda resultante de la co-simulación del NCO para tasa normal de transmisión (Normal-Rate) con un valor de entrada $A_{NR} = 16\,777$. La forma de onda superior representa la componente I de la señal y la forma de onda inferior representa la componente Q. Se agregaron dos marcadores para evaluar el período de la onda y verificar la frecuencia de la misma. La señal de validación no se muestra ya que la misma está todo tiempo en alto, señalizando que la salida del NCO siempre es válida habiendo tenido una entrada.

valida la implementación y prueba el funcionamiento para esta frecuencia de salida. De igual forma se implementó el valor de entrada negativo y se obtuvo el mismo conjunto de señales pero desfasadas en el sentido contrario.

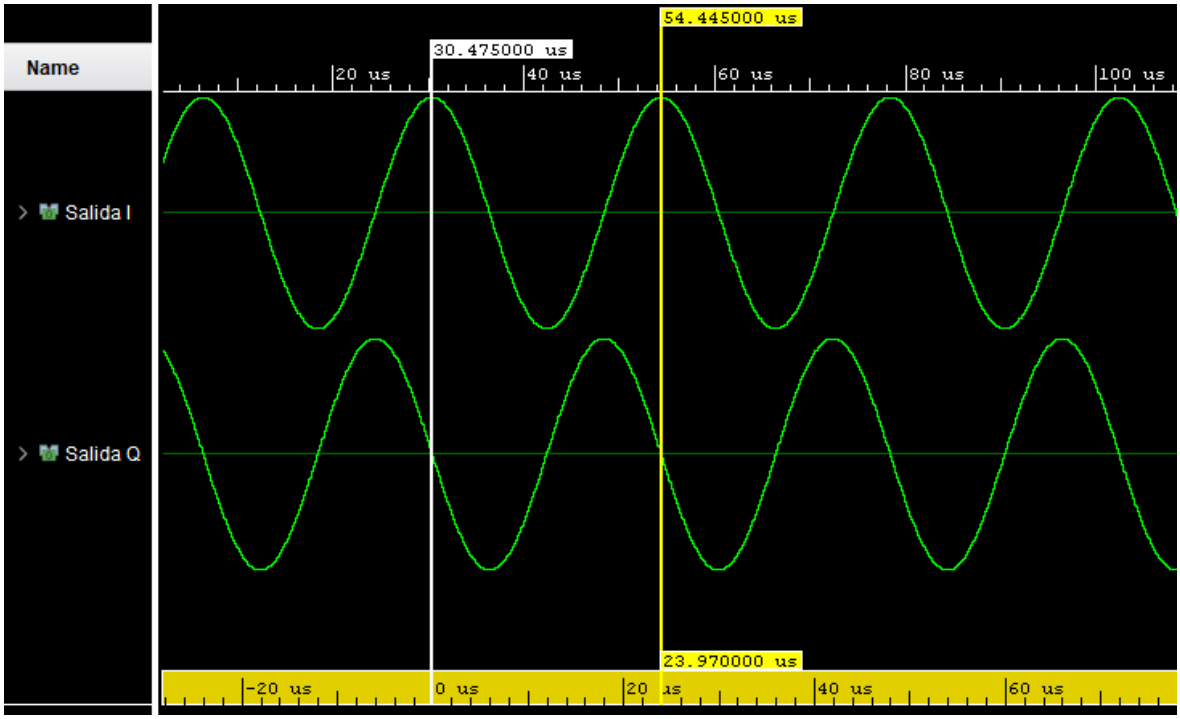


Figura 5.12: Forma de onda resultante de la co-simulación del NCO para tasa alta de transmisión (Hi-Rate) con un valor de entrada $A_{NR} = 13\,981$. La forma de onda superior representa la componente I de la señal y la forma de onda inferior representa la componente Q. Se agregaron dos marcadores para evaluar el período de la onda y verificar la frecuencia de la misma. La señal de validación no se muestra ya que la misma está todo tiempo en alto, señalizando que la salida del NCO siempre es válida habiendo tenido una entrada.

Se observa en la Figura 5.12 que el período de las formas de onda salientes es $T_{HR} \approx 23{,}97\,\mu s$, por lo que su frecuencia es muy parecida a la esperada de $f_{NR} = +41{,}667\,kHz$. Esto valida la implementación y prueba el funcionamiento para esta frecuencia de salida. De igual forma se implementó el valor de entrada negativo y se obtuvo el mismo conjunto de señales pero desfasadas en el sentido contrario.

Se observa en la Figura 5.13 que el período de las formas de onda salientes es $T_{LR} \approx 208{,}05\,\mu s$, por lo que su frecuencia es muy parecida a la esperada de $f_{NR} = +4{,}800\,kHz$. Esto valida la implementación y prueba el funcionamiento para esta frecuencia de salida. De igual forma se implementó el valor de entrada negativo y se obtuvo el mismo conjunto de señales pero desfasadas en el sentido contrario.

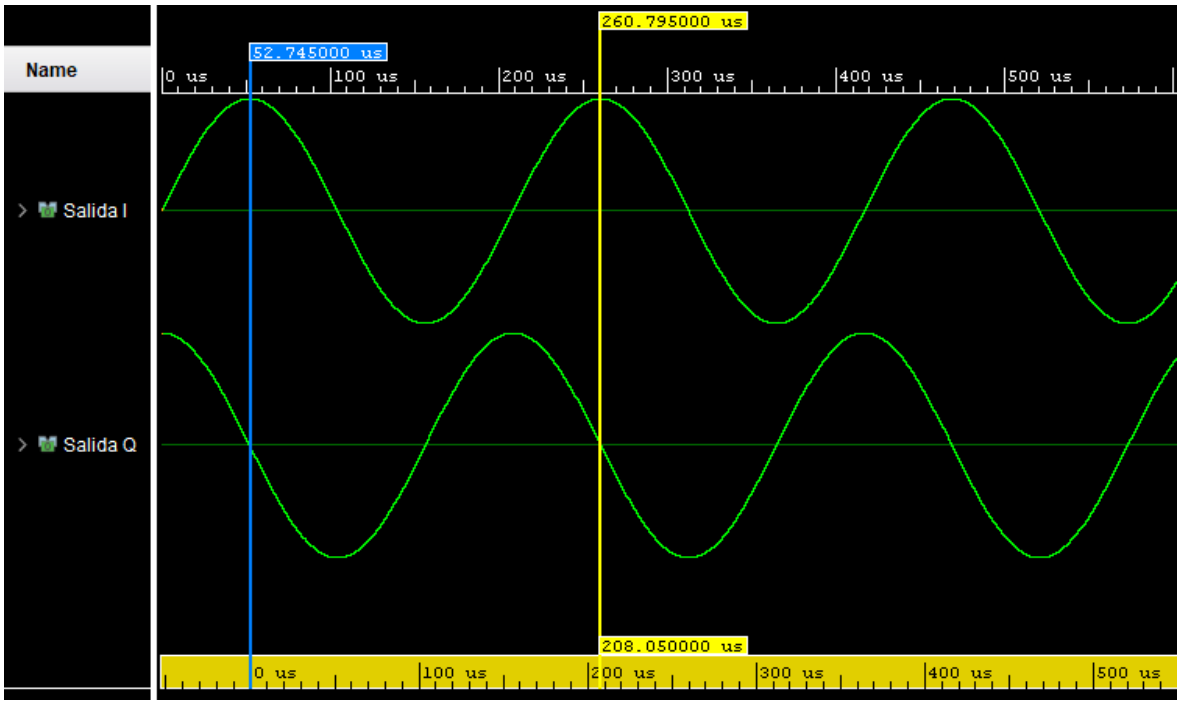


Figura 5.13: Forma de onda resultante de la co-simulación del NCO para tasa baja de transmisión (Low-Rate) con un valor de entrada $A_{NR} = 1611$. La forma de onda superior representa la componente I de la señal y la forma de onda inferior representa la componente Q. Se agregaron dos marcadores para evaluar el período de la onda y verificar la frecuencia de la misma. La señal de validación no se muestra ya que la misma está todo tiempo en alto, señalizando que la salida del NCO siempre es válida habiendo tenido una entrada.

5.6. Bloque Mezclador

El bloque Mezclador tiene como objetivo tomar la señal en banda base y llevarla a frecuencia intermedia f_{if} mediante la multiplicación de la misma por una portadora como se muestra en la Ecuación 5.6. Dado que la entrada son las componentes I y Q de la señal en banda base, para levantar correctamente la frecuencia de la señal a la frecuencia intermedia, hay que realizar la siguiente operación[18], donde S_{if} es la señal llevada a frecuencia intermedia, f_{if} es la frecuencia intermedia propiamente dicha, I_{in} y Q_{in} son las componentes I y Q de la señal entrante respectivamente:

$$S_{if} = I_{in} \cos(2\pi f_{if} t) - Q_{in} \sin(2\pi f_{if} t) \tag{5.6}$$

Esto se llevó a cabo utilizando una tabla de búsqueda o LUT (por sus siglas en inglés) como en el caso del NCO. A diferencia del caso anterior, para reducir la cantidad de recursos se representó la senoidal a frecuencia intermedia con 8 valores. Cada muestra entrante nueva es multiplicada por el valor siguiente en la senoidal, de esta forma se logra recorrer todo el período y se repite el proceso cada 8 muestras de entrada. Los valores elegidos para representar el seno se muestran en la Tabla 5.7.

Estos valores fueron calculados con la función seno de la librería matemática del

Tabla	Valor del seno
LUT_SIN[0]	0
LUT_SIN[1]	+0,707 031 25
LUT_SIN[2]	+1
LUT_SIN[3]	+0,707 031 25
LUT_SIN[4]	0
LUT_SIN[5]	−0,707 031 25
LUT_SIN[6]	−1
LUT_SIN[7]	−0,707 031 25

Tabla 5.7: Valores de la LUT utilizados para representar una señal senoidal y llevar la señal de entrada a frecuencia intermedia $f_{if} = f_{clock}/8$.

lenguaje C++. Los mismos son redondeados automáticamente por HLS al momento de la compilación a la resolución mínima, que en este caso son 12 bit(2 bit para la parte entera con signo y 10 bit para la parte decimal), por lo que no son exactamente igual al valor ideal.

Esta solución si bien es más eficiente en recursos, pero por otro lado está atada a la frecuencia de reloj del sistema. Ya que $f_{clock} = 100\,\text{MHz}$, la frecuencia intermedia será 8 veces menor, por lo tanto $f_{if} = 12{,}5\,\text{MHz}$.

Tanto las entradas como su salida se implementaron utilizando interfaces AXI4-Stream[16]. La salida de este módulo se dirige directamente hacia la etapa de RF, por lo que se considera muy conveniente tener una interfaz estándar para interactuar con elementos que podrían ser de desarrollo ajeno. La especificación de las interfaces se observa en la Tabla 5.8.

Señal (Puerto)	I/O	Tipo	Descripción
I_in	Entrada	`Bus AXI4-Stream`	Componente en fase (I) (16 bit con signo).
Q_in	Entrada	`Bus AXI4-Stream`	Componente en cuatratura (Q) (16 bit con signo).
if_out	Salida	`Bus AXI4-Stream`	Señal llevada a frecuencia intermedia (16 bit con signo).

Tabla 5.8: Interfaces del bloque mezclador.

Si se quisiera estandarizar la frecuencia intermedia de salida, como por ejemplo la frecuencia $f_{if} = 10{,}7\,\text{MHz}$, se podría llevar a cabo cambiando la frecuencia de reloj del sistema. Siguiendo con el ejemplo se podría llevar a $f_{clock} = 8f_{if} = 85{,}6\,\text{MHz}$ para el caso de $f_{if} = 10{,}7\,\text{MHz}$. Tal cambio en el diseño impactaría en el funcionamiento de

algunos bloques por lo que se debería realizar un análisis adecuado.

Resultados de pruebas

Para verificar que la tabla fue implementada correctamente y que con cada muestra nueva la multiplicación se realiza por el valor siguiente de la LUT, se ingresó un flujo constante de unos por la entrada correspondiente a la componente I, y un flujo constante de ceros por la entrada correspondiente a la componente Q. En la Figura 5.14 se observa el resultado de la co-simulación obtenido.

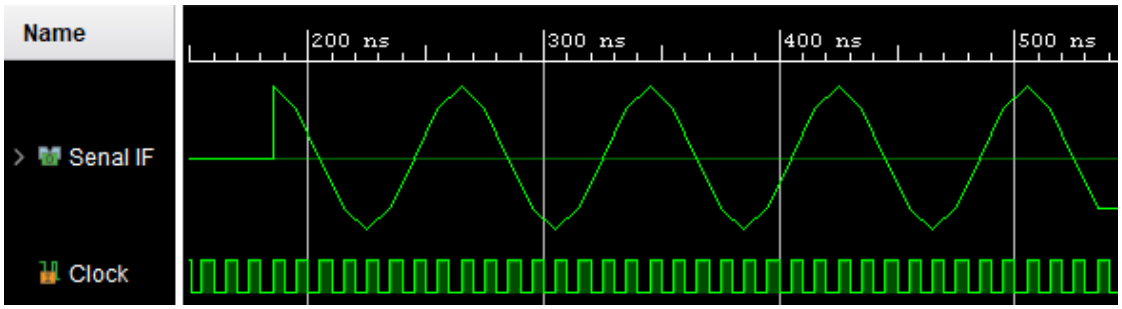


Figura 5.14: Co-simulación del Mezclador para una entrada de unos por la componente I y ceros por la componente Q.

Se puede ver que la señal de salida es un coseno con frecuencia $f_{if} = 12{,}5\,\text{MHz}$ como se esperaba por la Ecuación 5.6. La salida observada se corresponde con la LUT completa desplazada un cuarto de onda, recorrida una y otra vez en el orden correcto.

Luego se intercambiaron las entradas, por lo que se ingresó un flujo constante de ceros por la entrada correspondiente a la componente I, y un flujo constante de unos por la entrada correspondiente a la componente Q. En la Figura 5.15 se observa el resultado de la co-simulación obtenido.

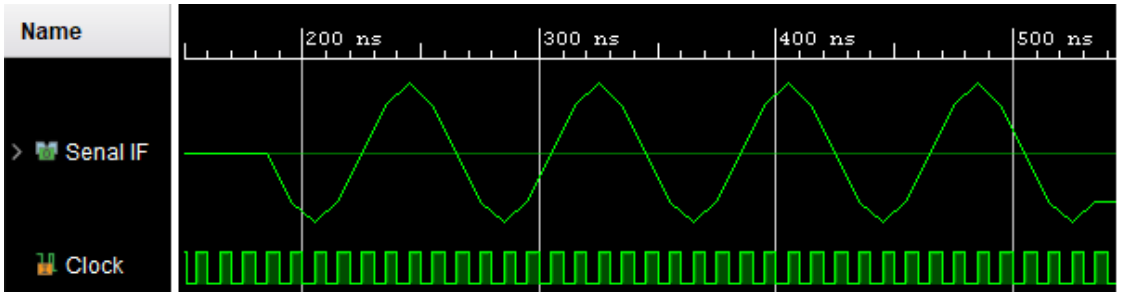


Figura 5.15: Co-simulación del Mezclador para una entrada de ceros por la componente I y unos por la componente Q.

Se puede ver que la señal de salida es un seno negativo con frecuencia $f_{if} = 12{,}5\,\text{MHz}$ como se esperaba por la Ecuación 5.6. La salida observada se corresponde con la LUT completa cambiada de signo, recorrida una y otra vez en el orden correcto.

Capítulo 6

Etapa de recepción

En el siguiente capítulo se muestra el modelo conceptual de la cadena completa de recepción y la implementación de cada bloque que la conforma. En primer lugar se discute el conjunto de elementos y a continuación se detalla cada uno de los bloques por separado con mayor detenimiento.

6.1. Modelo conceptual completo

El principio de funcionamiento básico de la cadena de recepción consiste en las siguientes acciones:

1. Ajustar la amplitud de la señal entrante.
2. Detectar la fase de la portadora en la frecuencia intermedia.
3. Demodular la señal.
4. Sincronizar la detección de símbolos.
5. Extraer los datos del paquete.
6. Decodificar los datos.

En la Figura 6.1 se muestra el esquema conceptual de la etapa de recepción diseñado con las especificaciones de D7A para realizar las tareas mencionadas.

El sistema completo posee dos entradas principales. La primera consiste en las muestras de la señal ya bajada a frecuencia intermedia provenientes del Front-End de radiofrecuencia (Front-End RF). La segunda es una señal que indica la potencia medida por el Front-End RF. La primera entrada está representada con 12 bit y la segunda con 8 bit. Al igual que en la etapa de transmisión, la frecuencia intermedia elegida para la señal de entrada fue $f_{if} = 12{,}5\,\text{MHz}$, pero con una frecuencia de muestreo de $f_s = 50\,\text{MHz}$.

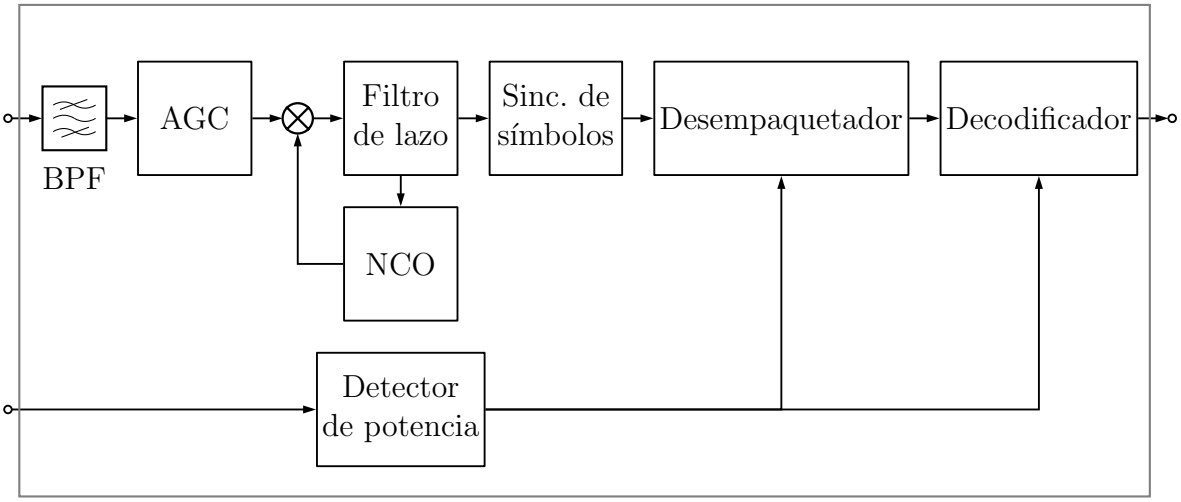


Figura 6.1: Esquema conceptual de la cadena de recepción completa. La

La salida del sistema consiste en los datos binarios en forma de bytes ya demodulados y decodificados, que son la entrada de la capa superior del protocolo. La cabecera del paquete, por otra parte, es descartada.

El primer bloque consiste en un filtro pasabanda (o BPF por sus siglas en inglés). Este es un elemento importante ya que su función es limitar el ancho de banda de ruido de la señal de entrada y filtrar componentes espurias fuera del área de interés. Además funciona como filtro anti alias para poder modificar la frecuencia de muestreo en los componentes siguientes. Sin este módulo, sería imposible realizar estos cambios en la frecuencia de muestreo, como se menciona más adelante.

El bloque siguiente en la cadena es el Control Automático de Ganancia o AGC (por sus siglas en inglés). Su función es ajustar la amplitud de la señal de entrada. Si bien la entrada de la cadena son muestras de 12 bit cada una, se consideró suficiente que la salida del AGC se represente con 8 bit simplemente, de forma tal de disminuir la cantidad de recursos necesarios para el resto del receptor. La importancia del módulo reside en que el protocolo está diseñado para comunicaciones inalámbricas. Esto implica que la amplitud de la señal recibida varía abruptamente dependiendo del canal. Sin esta funcionalidad es imposible realizar una comunicación de manera inalámbrica de forma estable.

Para la detección de fase y la demodulación se utiliza un Lazo de Seguimiento de Fase o PLL (por sus siglas en inglés)[18]. Este método incluye un lazo realimentado con un detector de fase, un filtro de lazo y un oscilador controlado. En el diseño se utilizó un multiplicador como detector de fase. El multiplicador y el filtro de lazo se implementaron de forma monolítica en un solo bloque aunque en la Figura 6.1 se muestren como dos elementos separados. Esta decisión se explica en el apartado correspondiente de este capítulo.

El NCO u oscilador controlado numéricamente es un bloque que toma un valor de

entrada y tiene como salida una señal sinusoidal de amplitud constante y frecuencia directamente proporcional al valor de la entrada, como se explicó en el capítulo anterior. Este módulo es la segunda parte del proceso de enganche de fase y demodulación. En particular se utilizó la misma implementación que se hizo para la etapa de transmisión. Es por este motivo que no se incluyó una sección en el capítulo sobre este módulo. Como se mencionó anteriormente, la salida del bloque consiste en dos señales sinusoidales que representan la componente en fase (Inphase o I) y la componente en cuadratura (Quadrature o Q) de un fasor que oscila a una frecuencia controlada, sin embargo sólo se utiliza la componente I para realimentar al multiplicador.

El bloque Sincronizador de símbolos tiene dos objetivos en particular. Por un lado implementa el conocido mecanismo de Early-Late Gate[19] para la sincronización de símbolo del sistema con la señal entrante. Por otra parte también se encarga de tomar una decisión sobre qué símbolo fue transmitido. Estas dos tareas están íntimamente relacionadas y por este motivo se incluyeron juntas en un solo bloque.

Una vez que se toma la decisión sobre qué bit fue transmitido, éste pasa al módulo Desempaquetador. Este bloque se encarga de recibir el flujo de bits provenientes del componente anterior e interpretarlos de alguna forma. Es decir, que en base a la estructura de paquete de la Figura 3.3, calcula la correlación que hay entre el flujo de bits recibidos y las diferentes palabras de sincronismo. Esto permite identificar el campo de datos para encaminarlos al siguiente bloque.

El Decodificador tiene como objetivo decodificar los datos y corregir errores si corresponde. La salida del bloque son los datos ya decodificados y listos para la capa superior del protocolo.

Por último, el Detector de potencia es un bloque que tiene la función de recibir un valor que indica cuánta potencia tiene la señal que recibe el Front-End de RF, representado con 8 bit. Si este valor supera un umbral especificado, se considera que hay una transmisión en curso y se levanta una bandera que activa los bloques Desempaquetador y Decodificador. Si el valor es menor que este umbral, se considera que no se está recibiendo nada o que la comunicación está cortada y se baja esta bandera. Esta señalización es necesaria debido a que los bloques Desempaquetador y Decodificador poseen variables internas que deben ser reiniciados con cada paquete nuevo.

Al igual que en el transmisor, todo el receptor podría haber sido implementado como un solo bloque usando HLS. Sin embargo se decidió no hacerlo de forma monolítica ya que se pretendía ver el valor de los datos intermedios que viajan de módulo en módulo y poder depurar el proceso más fácilmente. Además esta perspectiva agrega cierta modularidad que provee de claridad al modelo.

Por otro lado, las interfaces mencionadas (excepto aquellas que son parámetros de configuración o banderas) fueron implementadas utilizando el protocolo AXI4-Stream[16] para que las mismas sean estándar y el bloque sea reutilizable en otro

lugar sin necesidad de modificaciones. Esta decisión fue tomada debido a la simplicidad de indicarle a HLS el estándar de la interfaz. En adelante, este tipo de interfaces serán referenciadas como interfaces AXI4-Stream[16].

Para el diseño del sistema se utilizó una frecuencia de reloj $f_{clock} = 100\,\text{MHz}$. Todos los bloques utilizan la misma frecuencia.

6.2. Filtro pasabanda

Se diseñó un Filtro de Respuesta Finita al Impulso o Filtro FIR para llevar a cabo esta funcionalidad debido a que resulta muy fácil de programar y siempre es estable. Ya que la señal de interés está centrada en $f_{if} = 12{,}5\,\text{MHz}$ con un ancho de banda $B = 100\,\text{kHz}$ se diseñó el filtro de forma tal que su banda de paso se encuentre entre $f_{P1} = 12{,}45\,\text{MHz}$ y $f_{P2} = 12{,}55\,\text{MHz}$ y su banda de rechazo se encuentre a partir de $f_{S1} = 10{,}15\,\text{MHz}$ y $f_{S2} = 14{,}85\,\text{MHz}$. Se buscó también que la atenuación para las bandas de rechazo sea de al menos 40 dB.

El diseño obtenido fue un filtro FIR de 37 taps de tamaño. Se podría haber ideado un filtrado más angosto pero para eso la cantidad de taps, y por ende recursos necesarios para implementarlo, crece muy rápidamente. Esto se debe a que el factor de calidad necesario es muy alto debido a la gran tasa de muestreo que se posee y el ancho de banda angosto de la señal. Sin embargo, dada la frecuencia intermedia, resulta complicado disminuir la frecuencia de muestreo utilizada a la entrada del sistema.

Por otro lado se evaluó la degradación que tiene el filtro debido a la cuantización utilizada en la programación para verificar que los efectos no sean tan perjudiciales para el desempeño. La Figura 6.2 muestra el resultado obtenido del diseño.

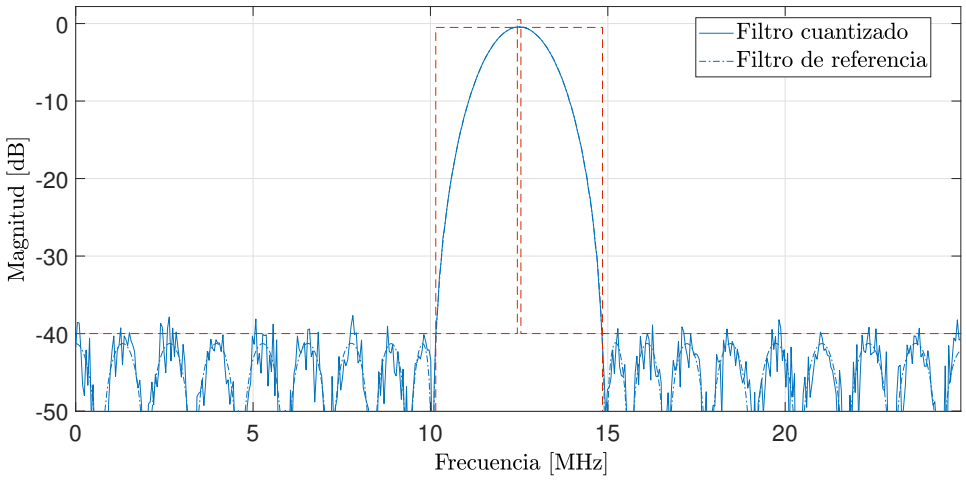


Figura 6.2: Respuesta en frecuencia del filtro pasabanda diseñado para el receptor.

En la Figura 6.3 se observa la respuesta al impulso del filtro diseñado.

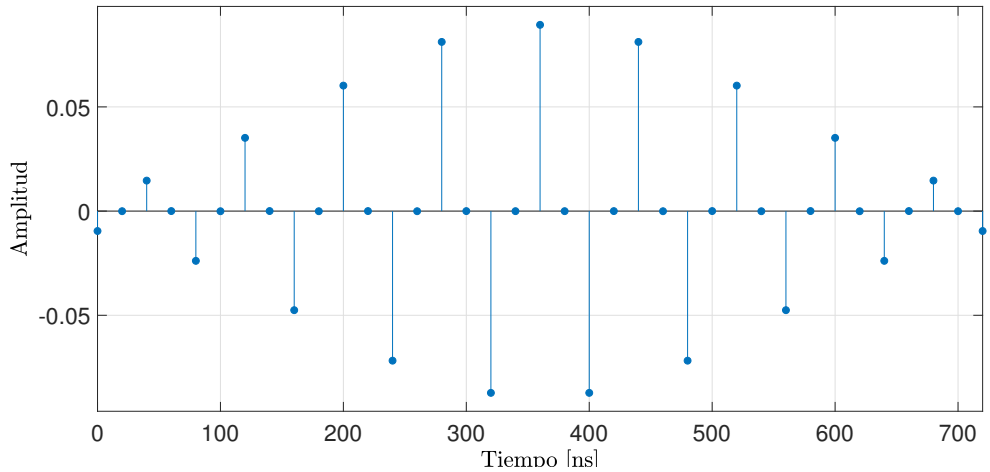


Figura 6.3: Respuesta al impulso del filtro pasabanda diseñado para el receptor.

Se puede ver que el efecto de la cuantización no modifica considerablemente el modelo, por lo que se consideró aceptable.

El bloque fue implementado utilizando una entrada y una salida. La entrada consiste en la señal en frecuencia intermedia $f_{if} = 12{,}5\,\text{MHz}$ a una tasa de muestreo de $f_s = 50\,\text{MHz}$ representada con 16 bit. La salida es la señal filtrada representada con 16 bit. Las interfaces se muestran en la Tabla 6.1.

Señal (Puerto)	I/O	Tipo	Descripción
signal_in	Entrada	`Bus AXI4-Stream`	Señal en $f_{if} = 12{,}5\,\text{MHz}$ (16 bit con signo).
signal_out	Salida	`Bus AXI4-Stream`	Señal filtrada (16 bit con signo).

Tabla 6.1: Interfaces del filtro pasabanda.

Resultados de pruebas

Para comprobar el funcionamiento del módulo, se introdujo un sólo pulso a la entrada y se verificó que la salida fuera la respuesta impulsiva correspondiente. El resultado de la co-simulación se observa en la Figura 6.4.

Se puede ver que la forma de onda se corresponde con la respuesta impulsiva de la Figura 6.4. Se verificó además que las amplitud de cada muestra de salida fuera acorde a la esperada. Se observa además que la salida tiene un retraso con respecto a la entrada. Esto se debe a que el módulo fue implementado con arquitectura pipeline para que sea capaz de recibir las muestras sin ninguna demora. Esta implementación fragmenta en etapas las operaciones a realizar, adicionando latencia al proceso total.

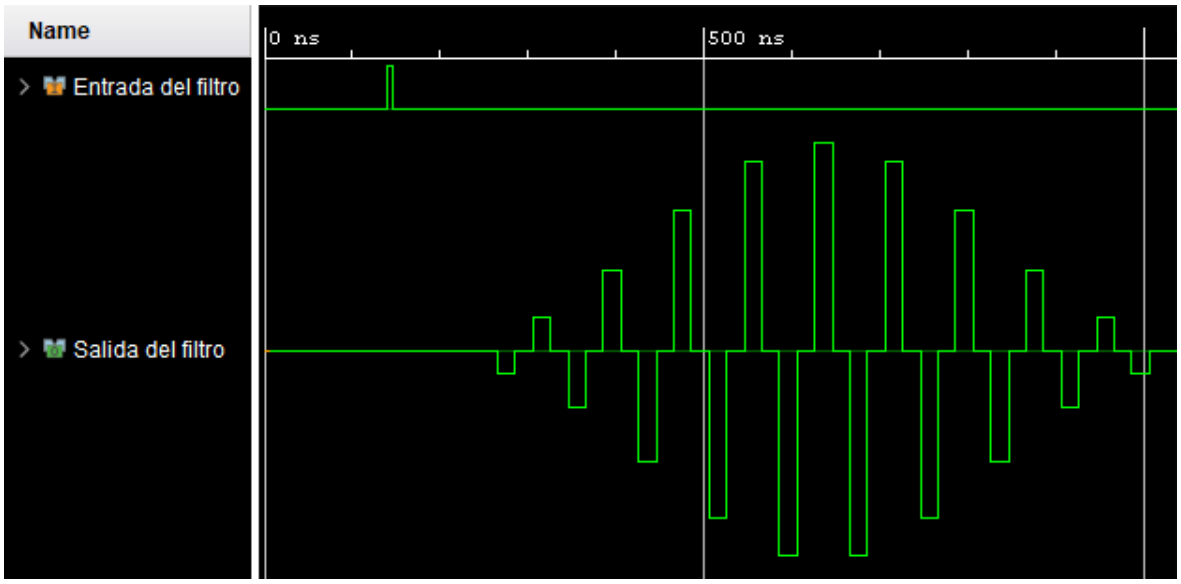


Figura 6.4: Co-simulación del filtro pasabanda ante un pulso de entrada.

6.3. Bloque de Control Automático de Ganancia

Existen muchas formas de implementar un componente que ajuste de forma automática la amplitud de la señal entrante. El funcionamiento básico de estas soluciones consiste principalmente en procesar la señal de entrada para estimar su potencia o un número proporcional a ella y ajustar de alguna forma la ganancia de la señal en base a este cálculo.

En el diseño propuesto no se estima la potencia de la señal, simplemente se toma el valor absoluto de la muestra entrante en cada momento para operar. Para no tener que almacenar un vector de muestras y poder ajustar esta ganancia en tiempo real se decidió implementar un cálculo de forma recursiva con cada muestra nueva.

Cada vez que ingresa una muestra nueva, esta operación calcula una amplitud ponderada de la señal en función del valor absoluto de la nueva muestra y el valor anterior de la amplitud ponderada. Esta ponderación se realiza mediante un factor de peso cuyo valor está entre 0 y 1. Específicamente, la operación realizada es la siguiente, donde A_n es la amplitud ponderada en el instante n, α es el factor de peso o factor de ponderación y x_{n+1} es la muestra en el instante $n+1$:

$$A_{n+1} = \alpha A_n + (1-\alpha)x_{n+1} \tag{6.1}$$

Una vez calculada la amplitud ponderada, se compara con una amplitud media deseada y la ganancia se calcula como la relación entre ambas. En la siguiente ecuación se observa el cálculo de la ganancia G en función de la amplitud media deseada A_d y la amplitud ponderada A_{n+1}:

$$G = \frac{A_d}{A_{n+1}} \tag{6.2}$$

El valor de la amplitud deseada debe ser determinado con ciertos cuidados en mente. En una señal senoidal, el valor que habría que seleccionar como A_d es el valor absoluto medio que se espera de la señal, el cual no es lo mismo que el valor pico.

Como no se utiliza la componente en fase y cuadratura para representar la señal d entrada, se espera que el valor calculado de la ponderación oscile en el corto plazo. Sin embargo, si la ponderación le da más peso a los valores anteriores que al nuevo (es decir, α muy cercano a 1), esta oscilación será relativamente poca.

Para darle más significado al factor α, se puede realizar algunas aproximaciones. Si $\alpha \approx 1$, con el correr de las muestras, repitiendo la Ecuación 6.1 se obtiene una exponencial decreciente que sigue la siguiente relación:

$$\Gamma = \alpha^n = e^{nln(\alpha)}$$

Por otro lado, la función de la recta tangencial a la función Γ en $n = 0$ tiene la forma:

$$P = \frac{d\Gamma}{dn}(0)\, n + \Gamma(0) \tag{6.3}$$

Si derivamos Γ con respecto a n, se obtiene:

$$\frac{d\Gamma}{dn} = ln(\alpha)e^{nln(\alpha)}$$

Para saber en qué valor de $n = \tau$ obtengo $P = 0$, con el resultado anterior se obtiene que $\frac{d\Gamma}{dn}(0) = ln(\alpha)$ y teniendo en cuenta que $\Gamma(0) = 1$, se sigue:

$$P = 0 = \tau ln(\alpha) + 1$$

Luego,

$$\tau = -\frac{1}{ln(\alpha)}$$

$$\alpha = e^{-\frac{1}{\tau}} \tag{6.4}$$

Por otra parte, también sabemos que el número de muestra n depende de la frecuencia de muestreo f_s y del tiempo t de la siguiente forma:

$$n = tf_s \tag{6.5}$$

Juntando las ecuaciones 6.4 y 6.5, sabiendo que $n = \tau$ se llega a:

$$\alpha = e^{-\frac{1}{tf_s}} \tag{6.6}$$

Esto permite encontrar un valor de ponderación α para obtener una constante de tiempo tiempo característico t a una frecuencia de muestreo f_s determinada.

Lo primero a tener en cuenta es que la solución desarrollada tiene una frecuencia de muestreo de la señal entrante igual a la mitad frecuencia de reloj del sistema (es decir, ingresa una muestra nueva por cada dos ciclos de reloj del sistema), luego $f_s = f_{clock}/2 = 50\,\text{MHz}$. Lo segundo es que se espera que la constante de tiempo t sea mucho más grande que el período de la portadora en frecuencia intermedia $T_{if} = 80\,\text{ns}$. Se buscó una constante de tiempo vinculada al tiempo de símbolo en tasa de transmisión normal $T_{sym} = 18\,\mu\text{s}$. En particular se obtuvo $t = T_{sym}/4 = 4{,}5\,\mu\text{s}$. Para este conjunto de factores, se obtuvo que:

$$\alpha \approx 0{,}996 \tag{6.7}$$

De esta forma, en un tiempo de símbolo, la señal ya ajusto su amplitud de forma considerable.

Por otra parte para seleccionar el valor de amplitud deseado A_d, se consideró que la señal entrante es una senoidal y que la salida está representada con 8 bit (7 bit para el valor absoluto y 1 bit para el signo). Teniendo en cuenta el valor cuadrático medio de la señal saliente, se eligió $A_d = 76$ para que de esta forma, los valores pico de la señal estarán dentro del rango representable de la salida aprovechando casi toda la resolución.

El bloque fue implementado utilizando una entrada y una salida. La entrada consiste en la señal sin amplificar representada con 12 bit. La salida es la señal amplificada representada con 8 bit. Las interfaces se muestran en la Tabla 6.1.

Señal (Puerto)	**I/O**	**Tipo**	**Descripción**
input_signal	Entrada	`Bus AXI4-Stream`	Señal a amplificar (12 bit con signo).
output_signal	Salida	`Bus AXI4-Stream`	Señal amplificada (8 bit con signo).

Tabla 6.2: Interfaces del AGC.

Resultados de pruebas

Una vez desarrollado el módulo, la primer prueba realizada fue ingresando un valor constante al bloque y verificando que la amplitud de la salida tienda a establecerse en A_d. Esta fue una prueba funcional a alto nivel para corroborar que las representaciones de las variables intermedias dentro del AGC fueran suficientes y el bloque hiciera lo esperado.

Para la prueba se ingresaron los valores A_{in}=2048, A_{in}=1024, A_{in}=512, A_{in}=256, A_{in}=128, A_{in}=64, A_{in}=32 y A_{in}=16. Se observó la tendencia de la señal de salida para cada entrada. El resultado de la simulación observa en la Figura 6.5.

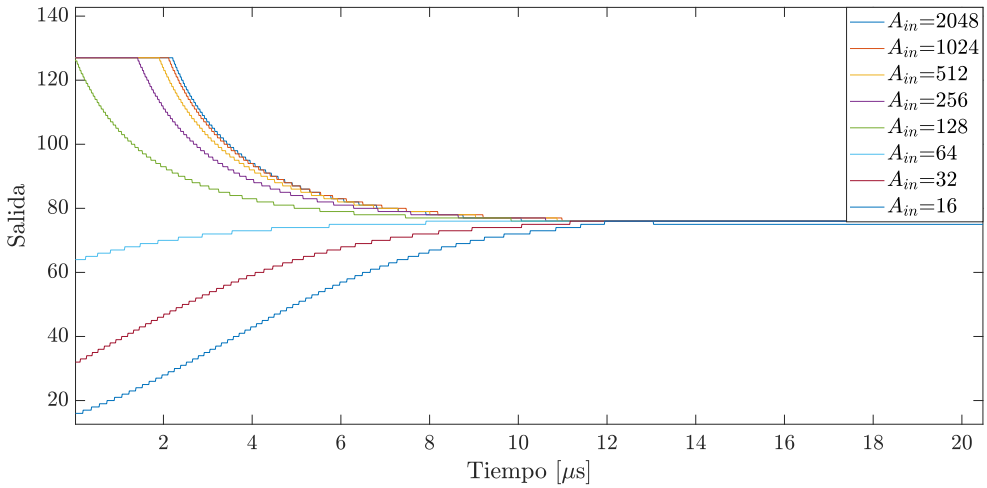


Figura 6.5: Resultados de simulación del AGC para valores de entrada constantes. Se ingresaron los valores A_{in}=2048, A_{in}=1024, A_{in}=512, A_{in}=256, A_{in}=128, A_{in}=64, A_{in}=32, A_{in}=16.

En la Figura 6.5 se puede ver como para cada caso la señal de salida evoluciona hasta estabilizarse en $A_d = 76$ o muy cerca del valor. Sólo en un caso se estabilizó en el valor 75. Esta diferencia se debe a que las variables dentro del módulo no tienen la suficiente resolución como para llegar al valor deseado en todos los casos.

Otra observación que se puede hacer es que las señales que son mayores que la representación máxima de salida saturan en el valor máximo alcanzable.

Por otra parte, se comprobó mediante simulaciones que para entradas con amplitud muy baja ($A_{in} \leq 8$) la evolución de la señal de salida no llega a estabilizarse en el valor deseado.

Se puede verificar en la figura que el tiempo en el que se estabiliza la señal de salida está del orden de un tiempo de símbolo $T_{sym} = 18\,\mu$s para una tasa de transmisión normal como era esperado. Este comportamiento garantiza que variaciones en la amplitud de la señal sean modificadas en un tiempo aceptable para el sistema.

Para verificar que el módulo funcione correctamente con una entradas senoidales, se realizó una prueba con una señal de entrada igual a un seno de amplitud 32 y frecuencia 1,25 MHz. El resultado de esta prueba funcional observa en la Figura 6.6.

Se puede ver que la amplitud máxima que alcanza la senoidal de salida se corresponde con el valor medio elegido $A_d = 76$.

Por otro lado, el tiempo de estabilización es igual al de la prueba anterior, por lo que se consideró aceptable para una implementación real.

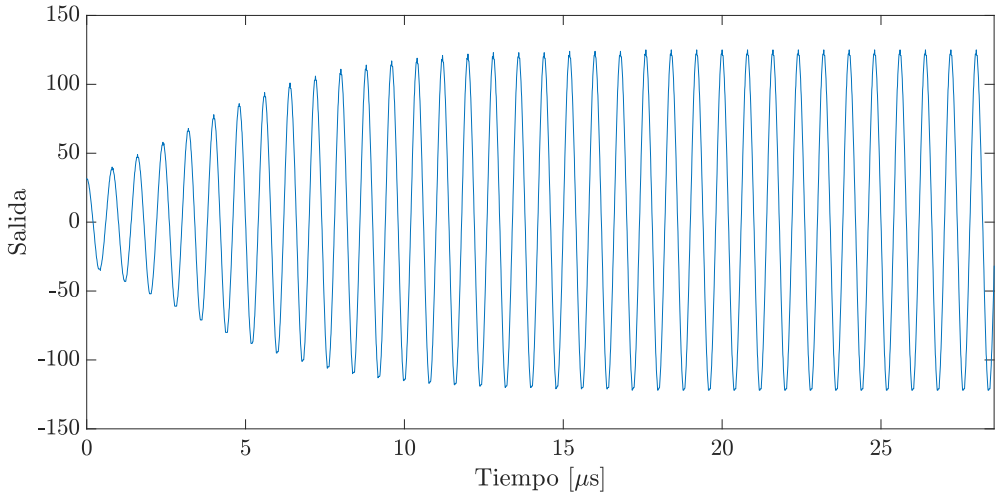


Figura 6.6: Resultados de simulación del AGC para una entrada senoidal de amplitud 32 y frecuencia 1,25 MHz.

6.4. Demodulador FSK basado en PLL

El primer elemento del Lazo de Seguimiento de Fase o PLL desarrollado es un multiplicador que realiza el producto entre la salida del NCO y la salida del AGC[18]. El segundo elemento es un filtro que tiene como objetivo procesar el producto mencionado y convertirlo en un valor proporcional a la diferencia de fase que existe entre las salidas del AGC y del NCO.

Para verificar el funcionamiento del PLL y ajustar los parámetros del filtro de lazo se realizó la simulación del PLL completo. En la Figura 6.7 se puede ver el esquema del PLL implementado.

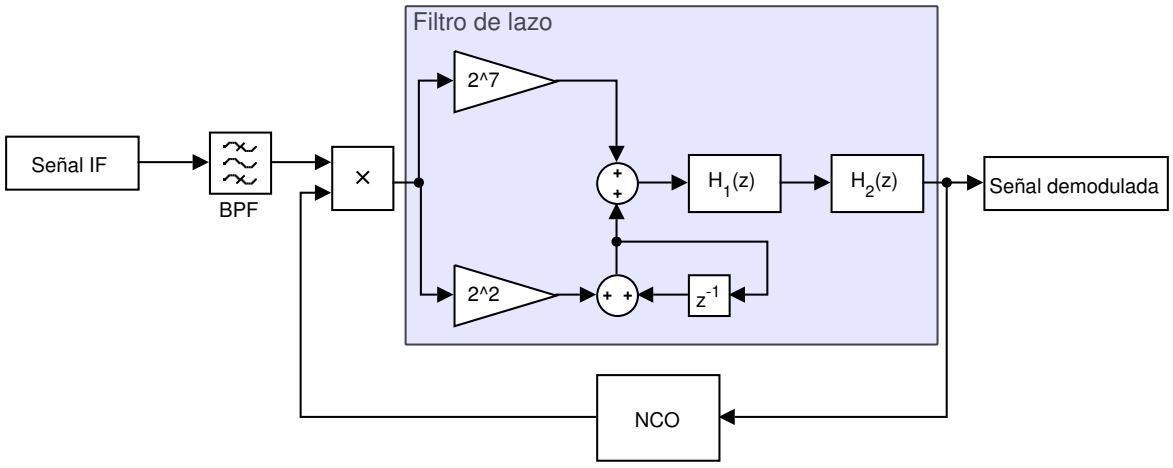


Figura 6.7: Diagrama en bloques del PLL utilizado en la cadena de recepción. El bloque llamado *Señal IF* simboliza la señal entrante en frecuencia intermedia. Se incluyó el filtro pasabanda (Sección 6.2) al diagrama debido a que su presencia es importante para el diseño. El bloque llamado *Señal demodulada* simboliza la señal ya demodulada que sale del lazo. Los bloques $H_1(z)$ y $H_2(z)$ son filtros pasabajos implementados como filtros de media móvil[20].

La tarea principal de un filtro de lazo es establecer el rango dinámico de funcionamiento del bucle y controlar la frecuencia de NCO. En esta aplicación del PLL se requiere un filtro que sea capaz de llevar a cero, no solo el desfasaje entre las señales del AGC y del NCO, sino también el seguimiento de las desviaciones de frecuencia dentro de un rango determinado. Este esquema permite extraer la señal en banda base en la salida del filtro de lazo a partir de estas desviaciones de frecuencia.

Para llevar a cabo esta funcionalidad se utilizó un filtro de lazo proporcional e integral (P+I en la jerga de teoría de control). El término proporcional consiste en una ganancia simple, por lo que a la salida del filtro esta rama contribuye con una señal que es proporcional a la entrada del filtro. El término integral consiste en un integrador ideal con una ganancia, por lo que a la salida del filtro esta otra rama contribuye con una señal que es proporcional a la integral de la entrada. Este término de acumulación es necesario para que el error de estado estacionario a la salida del PLL sea cero en presencia de un desplazamiento de frecuencia[21].

Por otra parte, se diseñó el multiplicador para tomar una muestra de cada cinco, tanto de la señal en frecuencia intermedia como de la senoidal proveniente del NCO. Aprovechando que el espectro fue filtrado por el filtro pasabanda inicial, este tipo de muestreo pasabanda permite al filtro de lazo operar con una frecuencia de muestreo mucho menor a la que poseen las etapas anteriores. Esto permite tener filtros con restricciones más relajadas y menos demandantes de recursos.

Se logró que cada rama del filtro de lazo tuviera una ganancia que fuera potencia de dos, para que de esta forma la implementación fuera muchísimo más sencilla en términos de recursos necesarios y programación.

Los bloques $H_1(z)$ y $H_2(z)$ referenciados en la Figura 6.7 son filtros pasabajos implementados como filtros de media móvil[20], cuyas funciones de transferencia son las que se muestran el la Ecuación 6.8 y Ecuación 6.9 respectivamente.

$$H_1(z) = \frac{1}{8} + \frac{1}{8}z^{-1} + \frac{1}{8}z^{-2} + \frac{1}{8}z^{-3} + \frac{1}{8}z^{-4} + \frac{1}{8}z^{-5} + \frac{1}{8}z^{-6} + \frac{1}{8}z^{-7} \tag{6.8}$$

$$H_2(z) = \frac{1}{4} + \frac{1}{4}z^{-1} + \frac{1}{4}z^{-2} + \frac{1}{4}z^{-3} \tag{6.9}$$

Lo que representa cada una de estas funciones es un filtro pasabajos que básicamente realiza un promedio de las ocho últimas muestras para el caso de $H_2(z)$ y cuatro últimas para $H_1(z)$. Es una forma poco eficiente pero extremadamente sencilla de implementar este tipo de funcionalidad. Haber elegido un coeficiente y una cantidad de muestras potencia de dos optimiza la cantidad de recursos que se requiere para realizar la operación de división, ya que esta puede realizarse desplazando bits solamente.

El primer tipo de simulación se realizó ingresando una señal a una frecuencia intermedia $f_{if} = 12{,}5\,\text{MHz}$ sin corrimiento de frecuencia pero con diferente fase θ y verificar

que el PLL se enganchara correctamente. Las fases utilizadas fueron $\theta = 0$, $\theta = \frac{1}{2}\pi$, $\theta = \pi$ y $\theta = \frac{3}{2}\pi$. El resultado de las simulaciones se puede ver en la Figura 6.8.

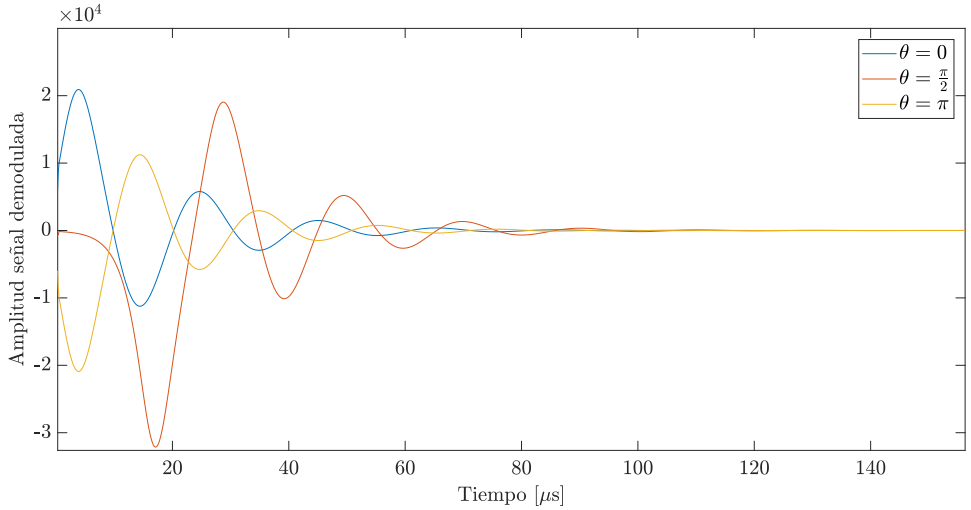


Figura 6.8: Simulación del PLL para una frecuencia intermedia $f_{if} = 12{,}5\,\text{MHz}$ pero con diferentes fases θ.

Se puede ver en la Figura 6.8 que para cada fase ingresada el PLL se enganchó correctamente. Además de esto el tiempo necesario en todos los casos fue menor a 120 µs.

El segundo tipo de simulación se realizó ingresando una señal a una frecuencia intermedia $f_{if} = 12{,}55\,\text{MHz}$, es decir con un corrimiento de 50 kHz sobre la frecuencia natural del sistema, con diferente fase θ y verificar que el PLL se enganchara correctamente. Este desplazamiento en frecuencia corresponde a un símbolo 1 de una tasa normal de transmisión (Tabla 3.2). Las fases utilizadas fueron $\theta = 0$, $\theta = \frac{1}{2}\pi$, $\theta = \pi$ y $\theta = \frac{3}{2}\pi$. El resultado de las simulaciones se puede ver en la Figura 6.9.

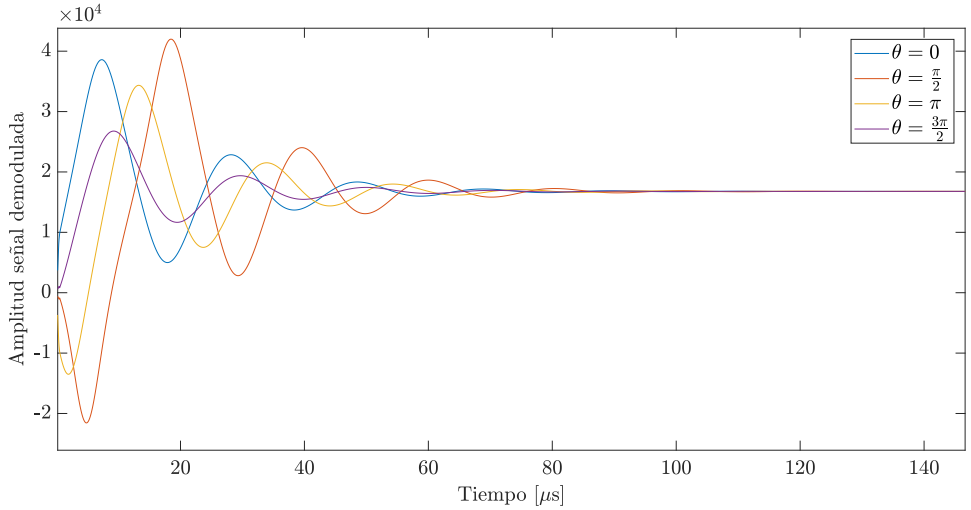


Figura 6.9: Simulación del PLL para una frecuencia intermedia $f_{if} = 12{,}55\,\text{MHz}$ y con diferentes fases θ.

El PLL se sincronizó con la fase y la frecuencia de entrada ya que la señal demodulada se estabilizó en un valor determinado, que corresponde a la entrada del NCO necesaria para alcanzar el desplazamiento en frecuencia. En todos los casos el tiempo necesario para que el error de fase sea aceptable fue menor a 120 µs. Aún para la tasa de transmisión alta ((Tabla 3.2)) y con el menor preámbulo posible (Tabla 3.1), el tiempo necesario para la sincronización es menor que el tiempo mínimo de preámbulo que permite el protocolo que es de 192 µs. Además puede verse en la Figura 6.9 que el valor de estabilización al que tiende el PLL es el mismo valor encontrado para que el NCO alcance la frecuencia del símbolo 1 encontrada en el capítulo anterior (Ecuación 5.3).

El otro tipo de simulación realizado y quizás el más concluyente, fue introducir un paquete generado por el transmisor ya llevado a frecuencia intermedia y comprobar que el modelo funciona. El paquete fue introducido con una fase aleatoria. La señal demodulada obtenida se observa en la Figura 6.10.

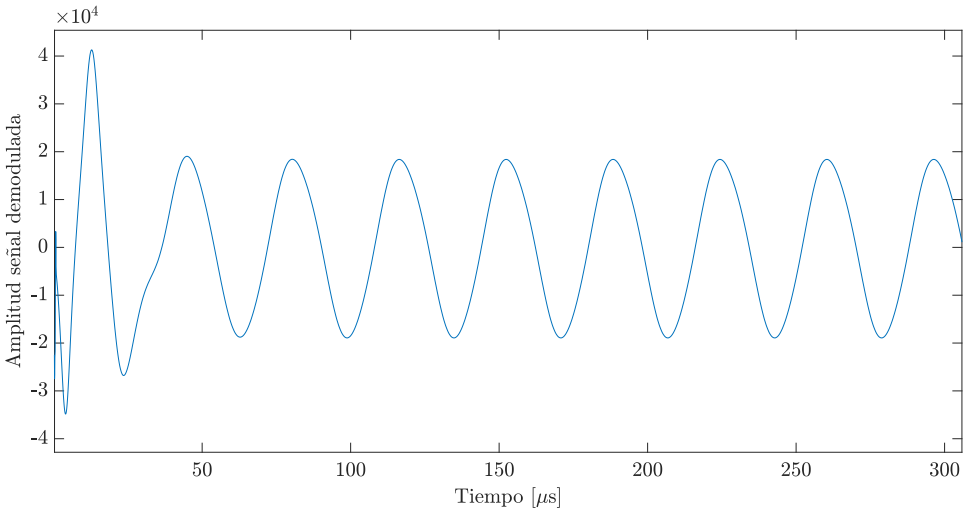


Figura 6.10: Simulación del PLL con un paquete real generado por el transmisor. En la figura sólo se muestra el preámbulo de forma de visualizar mejor el enganche y la demodulación. El paquete corresponde a una transmisión normal (Normal-Rate) según la Tabla 3.2.

En la Figura 6.10 se puede visualizar solamente una parte del preámbulo demodulado para que se vea más claramente cómo responde el sistema. Se observa que en cuestión de algunos símbolos el PLL ya se encuentra sincronizado con la señal de entrada y demodulando la misma para obtener la señal en banda base. Este funcionamiento es aceptable ya que permite una correcta demodulación del paquete con un margen de tiempo suficiente para que el siguiente módulo logre una buena sincronización de símbolo.

En primera instancia estas funcionalidades fueron implementadas de manera separada en diferentes bloques. Sin embargo, la latencia que implicaba esto debido a la comunicación entre bloques hacía que el funcionamiento del lazo fuera degradado con-

siderablemente ya que la realimentación estaba muy atrasada con respecto a la señal de entrada proveniente del AGC. Sin embargo, al implementarse ambas partes de forma monolítica en un solo bloque, este problema fue solucionado.

Para que los siguientes módulos en la cadena sean más simples y tengan restricciones de tiempo más relajadas, se consideró mucho más fácil extraer una cantidad de muestras por símbolo menor que la que se utiliza en el filtro de lazo. Además la señal es recortada de forma tal de sólo quedarse con el signo de la misma. Este diseño reduce de forma considerable los recursos que se tienen que utilizar en el módulo siguiente. Se decidió que la salida fueran 30 muestras por pulso (en lugar de las 180 muestras por pulso para la tasa de transmisión normal). Se consideró que es una cantidad razonable de muestras para sincronizarse con los símbolos y tener una buena resolución en el tiempo. Resulta evidente que este esquema no es óptimo ya que la decisión de bit no se realizará teniendo en cuenta toda la energía de la señal. Sin embargo, se consideró que la disminución de recursos requeridos en las etapas siguientes es una cualidad deseable en un protocolo de este tipo.

Teniendo en cuenta todas estas restricciones de diseño, el resultado final fue un bloque que posee dos entradas y dos salidas. Por un lado las entradas consisten en la salida del módulo AGC con una representación de 8 bit y la salida del módulo NCO representada con 16 bit. Por otra parte, las salidas consisten en el incremento de fase que controla el NCO representada con 32 bit (de los cuales se utilizan sólo los 25 bit menos significativos como se explicó en la Sección 5.5) y la señal demodulada recortada representada con solo 1 bit. Todas estas interfaces fueron implementadas utilizando AXI4-Stream[16] para que las mismas tengan un formato estándar. Las interfaces se observan en la Tabla 6.3.

Señal (Puerto)	**I/O**	**Tipo**	**Descripción**
input_signal	Entrada	`Bus AXI4-Stream`	Señal a demodular (8 bit con signo).
nco_output	Entrada	`Bus AXI4-Stream`	Salida del NCO (16 bit con signo).
nco_inc	Salida	`Bus AXI4-Stream`	Incremento de fase del NCO (32 bit con signo).
output_signal	Salida	`Bus AXI4-Stream`	Señal demodulada (8 bit con signo).

Tabla 6.3: Interfaces del demodulador basado en PLL.

Resultados de pruebas

Una vez implementado se procedió a realizar la co-simulación del modelo ya sintetizado. Para esto se ingresaron datos provenientes de la etapa de transmisión implementada para poder probar el sistema con un paquete real generado. En la Figura 6.11 se puede ver el resultado obtenido de la co-simulación.

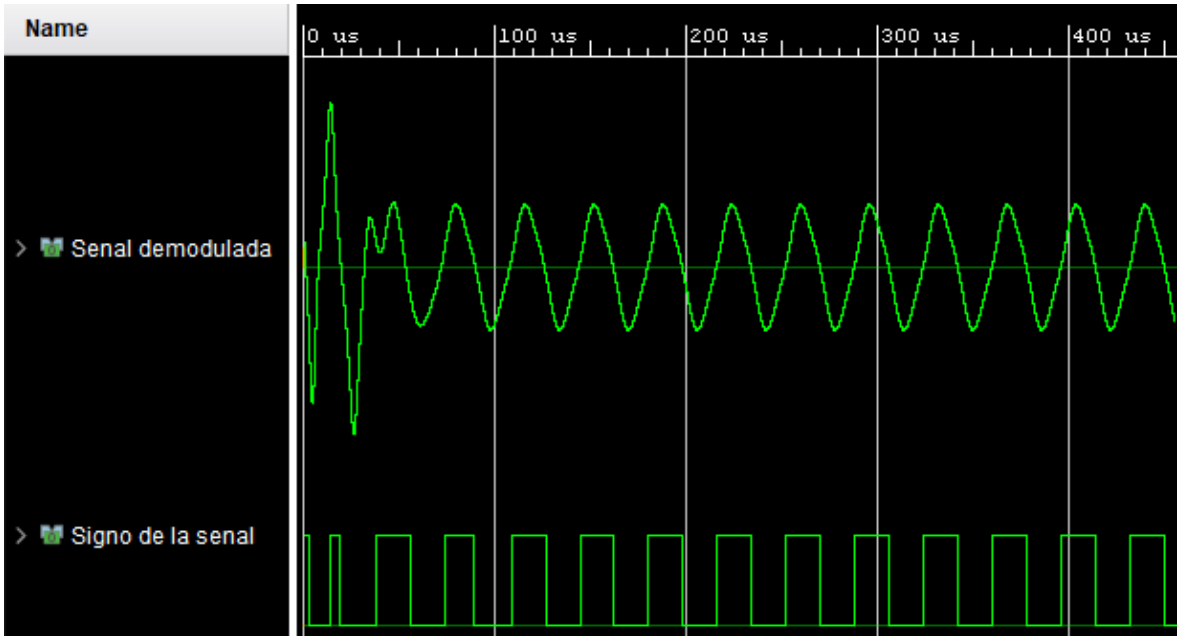


Figura 6.11: Co-simulación del modelo desarrollado de PLL. La forma de onda superior representa la salida del filtro de lazo (señal demodulada que es entrada del NCO). La forma de onda inferior representa la señal recortada que se encamina al siguiente módulo.

Se puede ver como el enganche sucede antes de que termine el preámbulo y haya tiempo suficiente para que el sincronizador de símbolo pueda funcionar correctamente. En la co-simulación, el preámbulo del paquete termina aproximadamente a los 730 µs de comenzada la misma. En el capítulo siguiente se puede observar la co-simulación de la demodulación de un paquete completo.

En la Figura 6.12 se puede verificar la forma del pulso saliente y la cantidad de muestras por cada uno de ellos.

Se puede ver que la cantidad de muestras por pulso es 30, por lo que el modelo implementado cumple con las indicaciones del diseño.

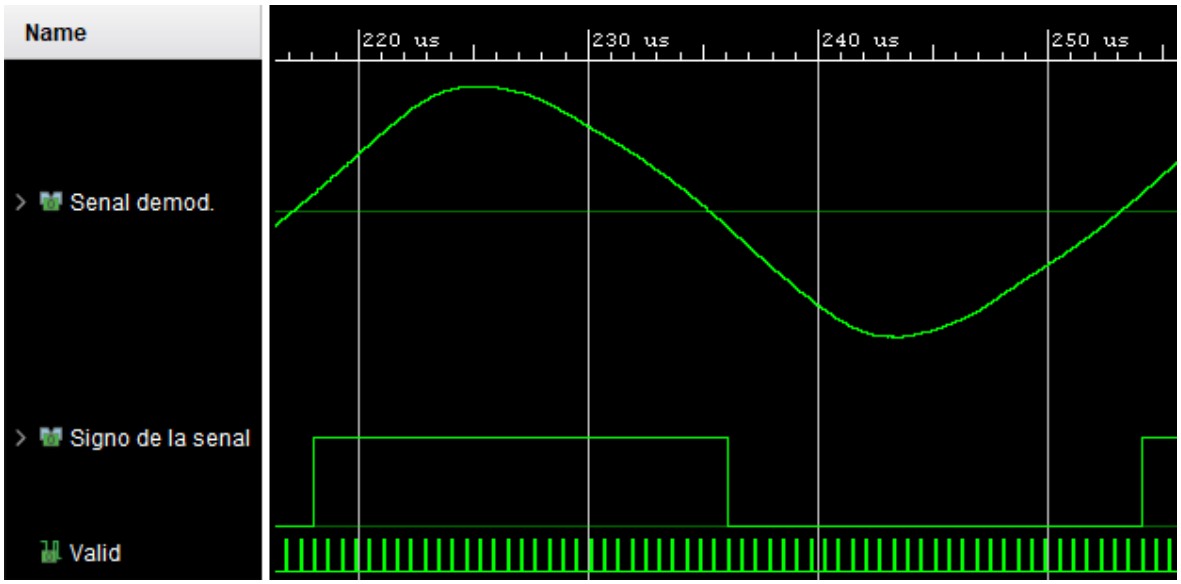


Figura 6.12: Ampliación de un símbolo en la co-simulación del modelo ya sintetizado del PLL. La señal *Valid* indica la bandera de validación de la muestra de la señal recortada. Las otras dos señales son las mismas que las de la Figura 6.11.

6.5. Sincronizador de símbolos

Existen muchos métodos de sincronización de símbolo que obtienen buenos resultados. Los más fáciles de implementar son aquellos a lazo abierto, en donde no existe realimentación de ningún tipo en la cadena de procesamiento. Sin embargo estos procedimientos tienen el gran problema del error de seguimiento con media distinta de cero. A diferencia de éstos, los sincronizadores a lazo cerrado por otra parte utilizan mediciones que controlan de qué forma el sistema se posiciona sobre las muestras de la señal, por lo que poseen un error de seguimiento de media cero. Por este motivo se decidió implementar el segundo tipo de sincronizador mencionado.

Entre los sincronizadores de lazo cerrado más conocidos se encuentra el mecanismo Early-Late Gate[19]. En la implementación realizada no se desarrolló exactamente el sincronizador clásico, sino que se hicieron algunas adaptaciones, pero en esencia el principio es el mismo. Es decir, se analiza un fragmento inicial de lo que se considera un símbolo y un fragmento de igual tamaño al final, y a partir de su comparación el sistema ajusta el sistema de referencia temporal utilizado.

La forma de emular el comportamiento típico del sincronizador Early-Late que se desarrolló utiliza un búfer de muestras correspondiente a un símbolo. El algoritmo almacena las muestras en el búfer de forma secuencial. Cuando el mismo se completa, realiza la integración de la ventana Early y de la ventana Late. En este caso, la ventana Early es el primer medio símbolo y la ventana Late es el segundo medio símbolo. El valor absoluto de los resultados son comparados y a partir de esto se desplaza la referencia temporal que tiene el sistema, y en el caso de tener que reutilizar muestras, las mismas

se mueven dentro del búfer.

El valor de ajuste en la fase a su vez es filtrado utilizando un filtro pasabajos, el cual realiza el promedio entre sus últimos valores obtenidos. Esto permite que el modelo sea más tolerante a perturbaciones, pero por otra parte el tiempo necesario para llegar a un estado estacionario es mayor. A medida que el promedio utiliza más valores anteriores, los efectos mencionados se intensifican. Es por esto que se implementó el sincronizador promediando las dos últimas muestras y las cuatro últimas muestras. A partir de los resultados obtenidos se eligió uno de ellos para el modelo final.

El bloque fue implementado utilizando una entrada y una salida. La entrada consiste en el bit de signo de las muestras de la señal demodulada representado con 8 bit debido a que se utilizó AXI4-Stream[16]. La salida es el bit por el cuál se tomó la decisión representado con 8 bit por el mismo motivo que en caso anterior. Las interfaces se muestran en la Tabla 6.1.

Señal (Puerto)	I/O	Tipo	Descripción
input_signal	Entrada	`Bus AXI4-Stream`	Bit de signo de la señal demodulada.
bitstream	Salida	`Bus AXI4-Stream`	Bit demodulado.

Tabla 6.4: Interfaces del sincronizador de símbolo.

Resultados de pruebas

El resultado obtenido de la co-simulación del sincronizador implementado utilizando un filtro que promedia los dos últimos valores se observa en la Figura 6.13.

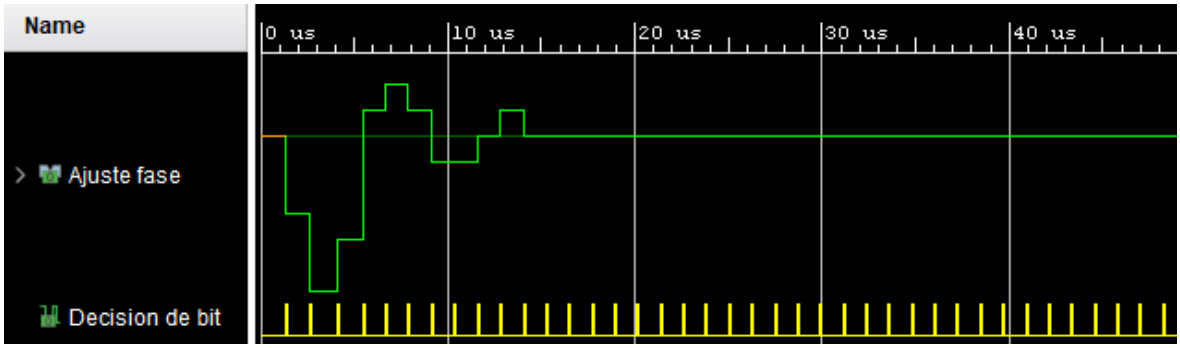


Figura 6.13: Co-simulación obtenida de la implementación del sincronizador de símbolo Early-Late con un filtro de dos valores. La señal *Ajuste fase* muestra el ajuste realizado por el bloque sincronizarse con el símbolo recibido. El valor mínimo que toma la señal es -6 y el máximo es 2. La señal *Decisión de bit* simboliza el temporizado de la decisión sobre que bit fue transmitido.

El resultado obtenido de la co-simulación del sincronizador implementado utilizando un filtro que promedia los cuatro últimos valores se observa en la Figura 6.14.

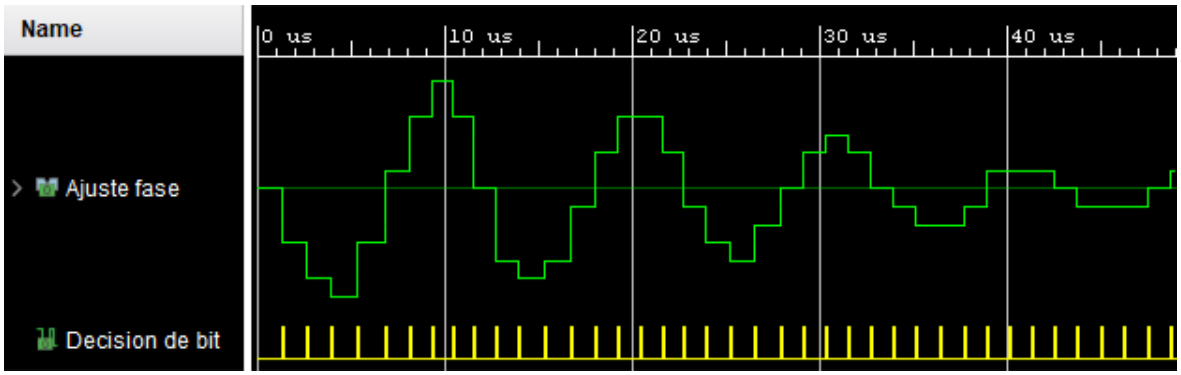


Figura 6.14: Co-simulación obtenida de la implementación del sincronizador de símbolo Early-Late con un filtro de cuatro valores. La señal *Ajuste fase* muestra el ajuste realizado por el bloque sincronizarse con el símbolo recibido. El valor mínimo que toma la señal es -5 y el máximo es 5. La señal *Decisión de bit* simboliza el temporizado de la decisión sobre que bit fue transmitido.

Como se esperaba, el ajuste realizado por el módulo que promedia dos valores calculados se estabiliza mucho más rápido que el módulo que promedia cuatro valores calculados. Sin embargo esta cualidad lo hace más sensible al ruido también.

De igual manera, se puede ver en la Figura 6.14 que la cantidad de símbolos que tiene que evaluar para que el sincronizador se estabilice es mayor a 40. Este diseño resulta imposible de implementar ya que la secuencia utilizada para la sincronización de símbolo en un paquete típico, posee 32 símbolos en total. Se comprobó que el diseño lleva a cabo una sincronización demasiado lenta para poder ser implementado. Una posible mejora es disminuir la ganancia del filtro de lazo para que esta oscilación sea menor y llegue al estado estacionario más rápido.

A partir del último resultado se procedió a probar el mecanismo pesando las últimas cuatro muestras pero con una ganancia menor. Es decir, ya no se realiza el promedio entre las últimas cuatro muestras, sino que se las suma y se divide el resultado por 8. De esta forma el filtro resultante tiene la transferencia que se muestra a continuación en la Ecuación 6.10.

$$H_2(z) = \frac{1}{8} + \frac{1}{8}z^{-1} + \frac{1}{8}z^{-2} + \frac{1}{8}z^{-3} \tag{6.10}$$

El resultado de la co-simulación de esta implementación se observa en la Figura 6.15.

Se puede ver que el tiempo de establecimiento y la oscilación es menor que en el caso del promedio de cuatro muestras. Con esta implementación se puede llegar a una sincronización en un margen de tiempo aceptable. Además tiene en cuenta las últimas cuatro muestras, haciendo el modelo menos susceptible al ruido. Por estos motivos, se decidió incluir este sincronizador al receptor.

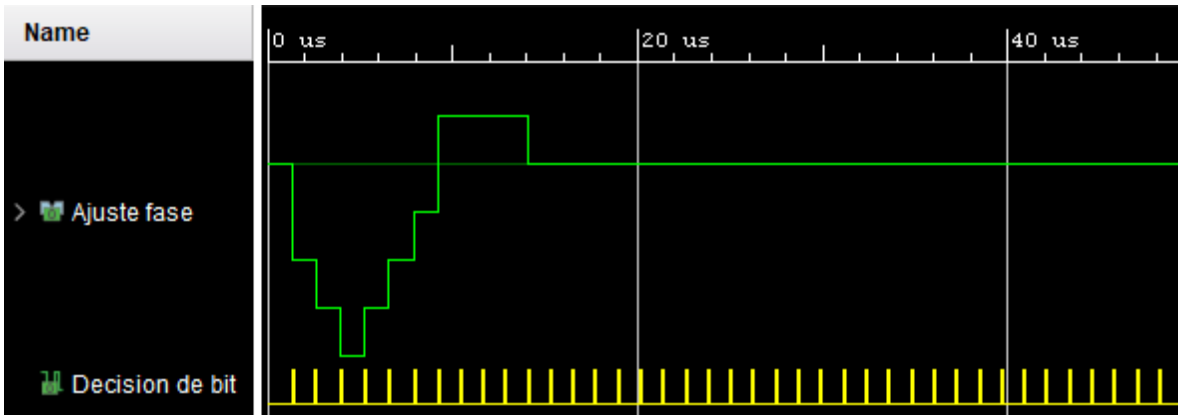


Figura 6.15: Co-simulación obtenida de la implementación del sincronizador de símbolo Early-Late con un filtro de cuatro valores con menor ganancia. La señal *Ajuste fase* muestra el ajuste realizado por el bloque sincronizarse con el símbolo recibido. El valor mínimo que toma la señal es -4 y el máximo es 2. La señal *Decisión de bit* simboliza el temporizado de la decisión sobre que bit fue transmitido.

6.6. Bloque Detector de potencia

Este módulo tiene la función de recibir un el valor de potencia recibido proveniente del Front-End de RF. A partir de esta entrada, el bloque simplemente levanta una bandera o la baja si la misma es mayor o menor a un umbral de referencia respectivamente. Se implementó un comportamiento de histéresis para mejorar el desempeño de forma tal que fluctuaciones cerca del umbral no causen reinicios problemáticos de los bloques correspondientes.

Este elemento se utiliza para indicar al Desempaquetador y al Decodificador que hay una recepción en curso. Esto le permite a estos módulos poder reestablecer sus variables y elementos internos cuando la bandera del detector de potencia está baja y poder procesar correctamente los datos de entrada cuando la bandera está alta.

Se definieron arbitrariamente dos valores de umbral cuyos valores son 100 y 50, correspondientes al umbral usado para cuando la potencia tiene pendiente positiva y cuando la potencia tiene pendiente negativa respectivamente. Estos valores pueden ser muy fácilmente cambiados a la hora de su implementación o programar que los mismos sean introducidos externamente. Sin embargo, como la etapa de RF no está dentro del alcance del proyecto, se decidió dejar el bloque implementado de la forma mencionada.

El bloque utiliza una entrada y una salida. Por un lado la entrada consiste en el valor de potencia representado con 8 bit (sin signo), y por otro, la salida consiste en 1 bit para indicar la recepción en curso. En ninguno de los casos se implementó un tipo de interfaz estándar ni señal de validación.

Señal (Puerto)	I/O	Tipo	Descripción
power_in	Entrada	`STD_LOGIC_VECTOR[7:0]`	Potencia recibida (8 bit sin signo).
packet_enable	Salida	`STD_LOGIC_VECTOR[0:0]`[1]	Señalización de recepción en curso: `1` → hay recepción `0` → no hay recepción

1 Tipo de interfaz establecido por HLS para entradas de 1 bit.

Tabla 6.5: Interfaces del detector de potencia.

Resultados de pruebas

Una vez sintetizado, se realizó la co-simulación del módulo para comprobar que el mismo funcione de manera correcta. Para ello se ingresó una señal que barre todas las posibles entradas del módulo y se verificó que cuando la misma cruza el umbral mayor, la bandera de salida se pone en 1, y cuando baja del umbral menor se pone en 0. En la Figura 6.16 se observa el resultado de esta co-simulación.

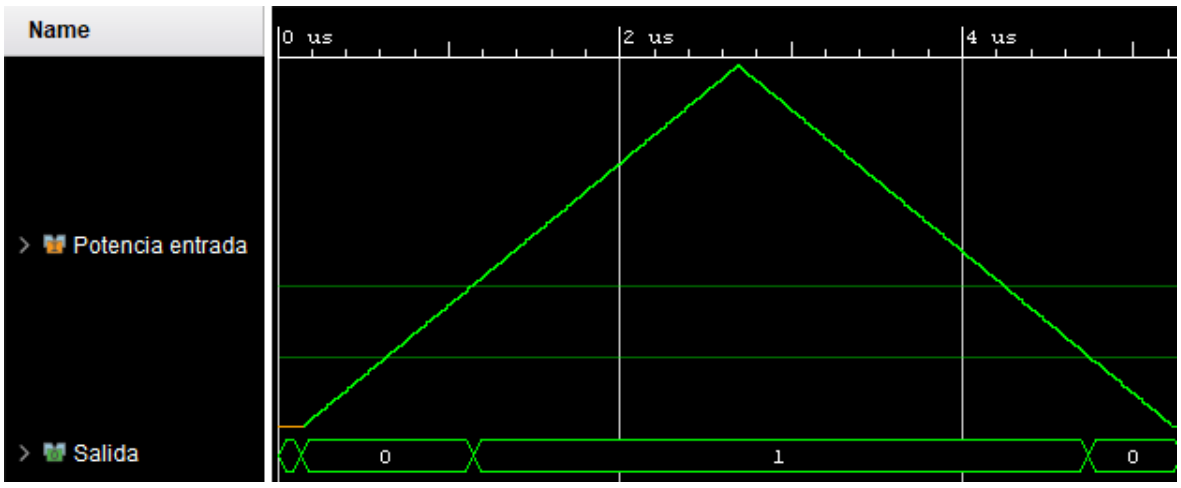


Figura 6.16: Co-simulación del detector de potencia. La forma de onda superior representa el valor de entrada que tiene el módulo. La forma de onda inferior representa la salida. La líneas horizontales marcan los umbrales utilizados.

Se puede ver que cuando la señal tiene pendiente positiva y cruza el primer umbral, la señal de salida no se modifica. Por el contrario cuando cruza el umbral mayor, la señal cambia a 1. En el caso que la pendiente de la señal de entrada es negativa, ocurre el caso inverso.

Esto valida el diseño y comprueba que el módulo tiene el funcionamiento esperado.

6.7. Bloque Desempaquetador

A partir del flujo de bits obtenido de la demodulación, el siguiente paso es extraer la carga útil. Para esto, el protocolo dispone la palabra de sincronismo dentro del paquete, por lo que la tarea se resume en buscar esta palabra dentro de los bits obtenidos y a partir de ese momento rescatar los datos (Figura 3.3).

Para detectar la palabra de sincronismo se utiliza el concepto de Distancia de Hamming[19], que se define como el número de elementos en el que difieren dos vectores binarios. Es decir, se almacenan los últimos 16 bit que ingresan al módulo y se comparan con todas las palabras de sincronismo (Tabla 3.6). De este proceso se cuentan la cantidad de diferencias entre los vectores (distancia de Hamming) y a partir de esto se toma una decisión. Si la distancia es menor o igual a un valor umbral especificado, se considera que la secuencia ingresada es una palabra de sincronismo y los bits siguientes ingresados se retransmiten como datos al siguiente módulo. Si la distancia es mayor a esta referencia, se espera al siguiente bit y se repite el proceso.

Se definió un valor umbral ya que permite cierta tolerancia a errores de transmisión dentro de la palabra de sincronismo. Esto permite que si hubo una cantidad de errores determinada en la palabra, todavía se siga interpretando como tal. El único problema que tiene este diseño es que si la comunicación es ruidosa, alguna parte del preámbulo puede ser alterada de forma tal que caiga en las palabras interpretadas como palabra de sincronismo. Esto generaría una interpretación completamente errónea del paquete. De todas formas, para las simulaciones de este apartado este umbral se dejó en cero (es decir que los bits ingresados tienen que ser exactamente igual a la palabra) ya que no se introdujo ruido.

El módulo posee dos entradas principales y una salida. Por un lado, las entradas son los bits de datos representados con 1 bit proveniente del sincronizador de símbolo y una bandera de habilitación que indica cuando hay una recepción activa en el sistema completo. Por otro lado la salida son 8 bit de datos codificados que van a parar a una cola o FIFO para desacoplar el flujo de datos entre este módulo y el módulo que sigue (Decodificador). Tanto la interfaz de entrada de datos como la de salida están implementadas utilizando AXI4-Stream[16] para estandarizarlas. Las interfaces se observan en la Tabla 6.6.

Resultados de pruebas

Para probar el funcionamiento del bloque, se ingresaron los bits correspondientes a un paquete definido por el protocolo (Figura 3.3) con un conjunto de datos conocido.

Se pudo verificar que a la salida los datos fueran los mismos que los que se ingresaron, en el orden correcto y sin ningún tipo de elemento adicional.

Señal (Puerto)	I/O	Tipo	Descripción
input_data	Entrada	`Bus AXI4-Stream`	Flujo de bits demodulados.
packet_enable	Entrada	`STD_LOGIC_VECTOR[0:0]`[1]	Señalización de recepción en curso: `1` → hay recepción `0` → no hay recepción
output_data	Salida	`Bus AXI4-Stream`	Carga útil extraída del paquete.

1 Tipo de interfaz establecido por HLS para entradas de 1 bit.

Tabla 6.6: Interfaces del bloque desempaquetador.

Esta prueba se realizó para diferentes tipos de datos y diferentes tamaños y en todos los casos el resultado fue el esperado.

6.8. Bloque Decodificador

Una vez extraídos los datos del paquete, se procede con la decodificación. Tanto el PN9 como el Entrelazador se implementaron copiando la lógica utilizada en la codificación debido a que el proceso es exactamente el mismo en ambos casos. Por otro lado, el decodificador del código convolucional se logró implementado un decodificador Viterbi.

La lógica del mismo no fue desarrollada sino que se utilizó un algoritmo ajeno[22]. Se realizó de esta forma debido a que la lógica tiene una complejidad grande y el código referenciado es abierto. Por otra parte, una de las ventajas de HLS es la reutilización de código de alto nivel. La implementación no fue directa, sino que realizaron adaptaciones semánticas del código debido a que no toda descripción de alto nivel es sintetizable por HLS.

Los datos de entrada siguen el flujo de decodificación de la forma que se indica en el diagrama de flujos de la Figura 6.17. Una vez que los datos ingresan al módulo y son procesador por el PN9, pueden tomar dos caminos como se puede ver en la figura.

El bloque utiliza tres entradas y una salida. Las entradas consisten en los datos extraídos del paquete en forma de bytes, el esquema de codificación (Tabla 3.4) representado con 2 bit y la señalización de recepción en curso representada con 1 bit. La salida consiste en los datos decodificados en forma de bytes. Tanto la interfaz de entrada de datos como la de salida están implementadas utilizando AXI4-Stream[16] para estandarizarlas. Las interfaces se observan en la Tabla 6.7.

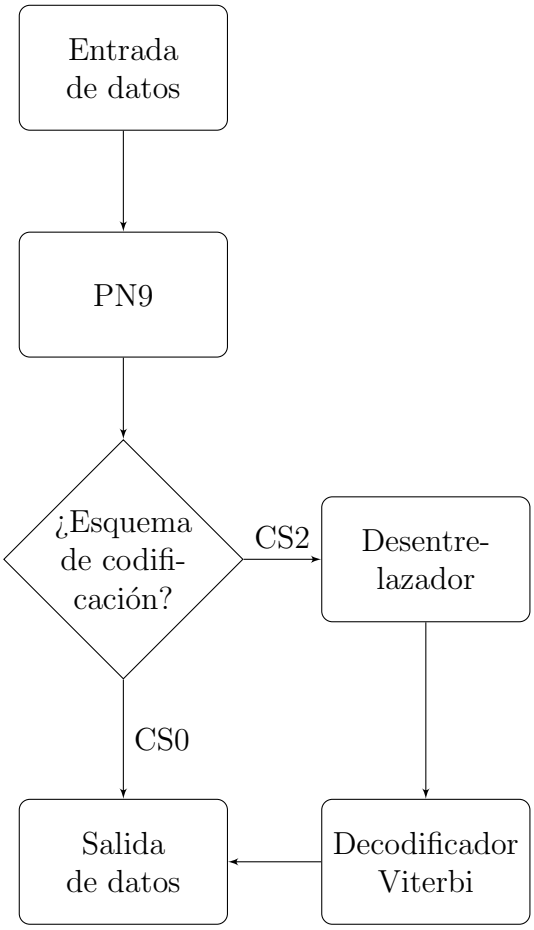


Figura 6.17: Diagrama de flujo del proceso de decodificación. CS0 y CS2 son esquemas de codificación especificados en la Tabla 3.4.

Señal (Puerto)	I/O	Tipo	Descripción
input_data	Entrada	`Bus AXI4-Stream`	Flujo de bits extraídos del paquete.
coding_scheme	Entrada	`STD_LOGIC_VECTOR[1:0]`	Esquema de codificación (Tabla 3.4)
packet_enable	Entrada	`STD_LOGIC_VECTOR[0:0]`[1]	Señalización de recepción en curso: `1` → hay recepción `0` → no hay recepción
output_data	Salida	`Bus AXI4-Stream`	Datos decodificados en forma de bytes.

1 Tipo de interfaz establecido por HLS para entradas de 1 bit.

Tabla 6.7: Interfaces del bloque decodificador.

Resultados de pruebas

Se realizó un seguimiento de las variables intermedias en la fase de pruebas funcionales de alto nivel y se pudo confirmar que el proceso era el correcto.

Por otro lado, utilizando datos codificados obtenidos de las simulaciones realizadas en la Sección 5.2, se realizó la decodificación de los mismos. Se logró obtener los datos originales sin codificar con éxito.

Para probar el decodificador Viterbi se utilizaron datos codificados generados con el codificador desarrollado en la Sección 5.2. El resultado obtenido fueron los datos originales introducidos al módulo de codificación.

Adicionalmente se probó introducir un bit erróneo en el flujo para comprobar que el decodificador realice la corrección del mismo. El resultado obtenido fueron los datos originales introducidos al módulo de codificación, por lo que se comprobó la corrección de errores del algoritmo.

Capítulo 7

Pruebas de integración

Una vez finalizada la implementación de cada bloque del transmisor y receptor, el siguiente paso consistió en realizar pruebas integradoras de todo el sistema. Las mismas consistieron en la co-simulación de la respectiva cadena de bloques. En este capítulo se discuten los resultados que se obtuvieron a partir del ensamblaje de todos los módulos. Por último se discuten las pruebas realizadas al receptor cuando la señal de entrada se encuentra sumergida en ruido gaussiano blanco aditivo.

7.1. Resultados obtenidos del transmisor

La primera etapa de validación del modelo fue verificar que exista una correcta comunicación entre el transmisor y el receptor sin la presencia de ruido. Esta prueba lo que demostraría es que en principio, el diseño conceptual funciona.

Cabe aclarar que esta prueba no garantiza que el diseño sea funcional en una implementación real debido a que no se tienen en cuenta factores externos, como el ruido.

En primer lugar se realizó todo el proceso de transmisión de un paquete completo. Se ingresaron al transmisor dos bytes de datos, los cuales fueron "`00010010 00110100`" o en formato hexadecimal "`0x1234`". Se utilizó el esquema de codificación CS0 (Tabla 3.4) y la clase de palabra de sincronismo 0 (Tabla 3.6).

Una vez codificados, los bytes de datos resultantes fueron "`11110011 00101001`" o en formato hexadecimal "`0xF329`". En la Figura 7.1 se observa la simulación que muestra este resultado.

En la Figura 7.1 se puede observar el intercambio de datos entre la entrada del sistema y el bloque codificador utilizando AXI4-Stream[16]. Cada ciclo de reloj que las señales *Valid* y *Ready* se encuentran en alto simultáneamente, ambas partes de la comunicación asumen que el dato fue transmitido correctamente. Se puede ver que esta operación sobre los datos binarios son en una escala temporal mucho menor que el procesamiento realizado por los bloques siguientes. El resto de la simulación se observa

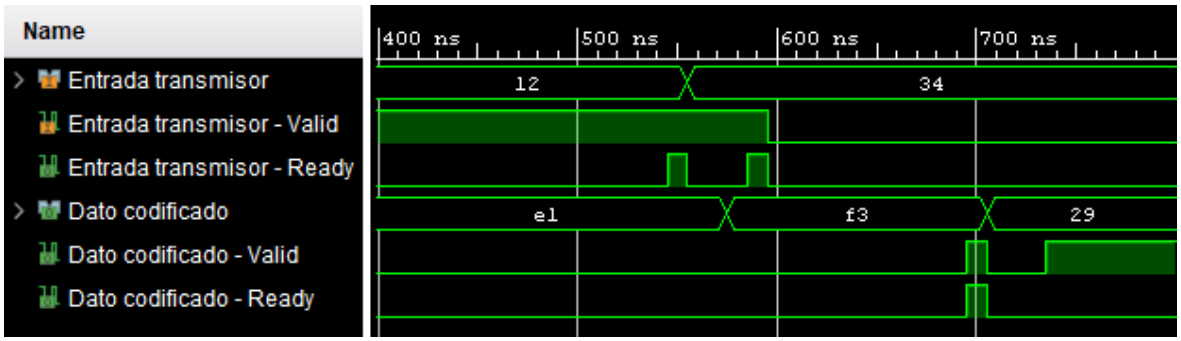


Figura 7.1: Co-simulación de la codificación de dos bytes de entrada. Las primeras tres formas de onda corresponden al intercambio AXI4-Stream[16] de los datos entre la entrada del sistema completo y el bloque codificador. Las últimas tres formas de onda corresponden al intercambio AXI4-Stream[16] entre de los datos ya codificados entre el bloque codificador y el bloque empaquetador.

en la Figura 7.2.

Se puede ver que el paquete fue correctamente generado y modulado en banda base según las especificaciones del protocolo. En la Figura 7.2 se marcó con líneas azules las partes del paquete para mejor interpretación. La primera parte corresponde a la rampa ascendente que está representada con la señal *Ganancia* de la figura.

El preámbulo es la segunda etapa consistiendo en una alternancia de unos y ceros, por lo que su señal en banda base son pulsos positivos y negativos respectivamente como se mencionó en la Sección 5.3. Esta representación se puede ver en la señal *Conformador Gaussiano* de la Figura 7.2.

La palabra de sincronización es la tercera etapa. En este caso la palabra transmitida en formato hexadecimal fue "`0xE6D0`"(Tabla 3.6) ya que se eligió el esquema de codificación CS0 y la clase de palabra de sincronismo 0. Se puede ver claramente los pulsos en la señal *Conformador Gaussiano* que se corresponden con su representación binaria "`111000110 11010000`".

La siguiente etapa corresponde a la carga útil que contiene los datos codificados. Ya que se utilizó el esquema de codificación CS0 (Tabla 3.4), los datos se procesaron utilizando PN9. Es por eso que los bytes de datos codificados resultantes fueron "`11110011 00101001`" o en formato hexadecimal "`0xF329`" como se mencionó anteriormente.

Por último la etapa de la rampa descendente representada con la señal *Ganancia* de la figura, al igual que la rampa ascendente. Esta etapa del paquete apaga de forma paulatina la transmisión del paquete.

No se incluyó la salida del mezclador ya que resulta poco ilustrativa una modulación a esa frecuencia con un ancho de banda relativamente tan bajo.

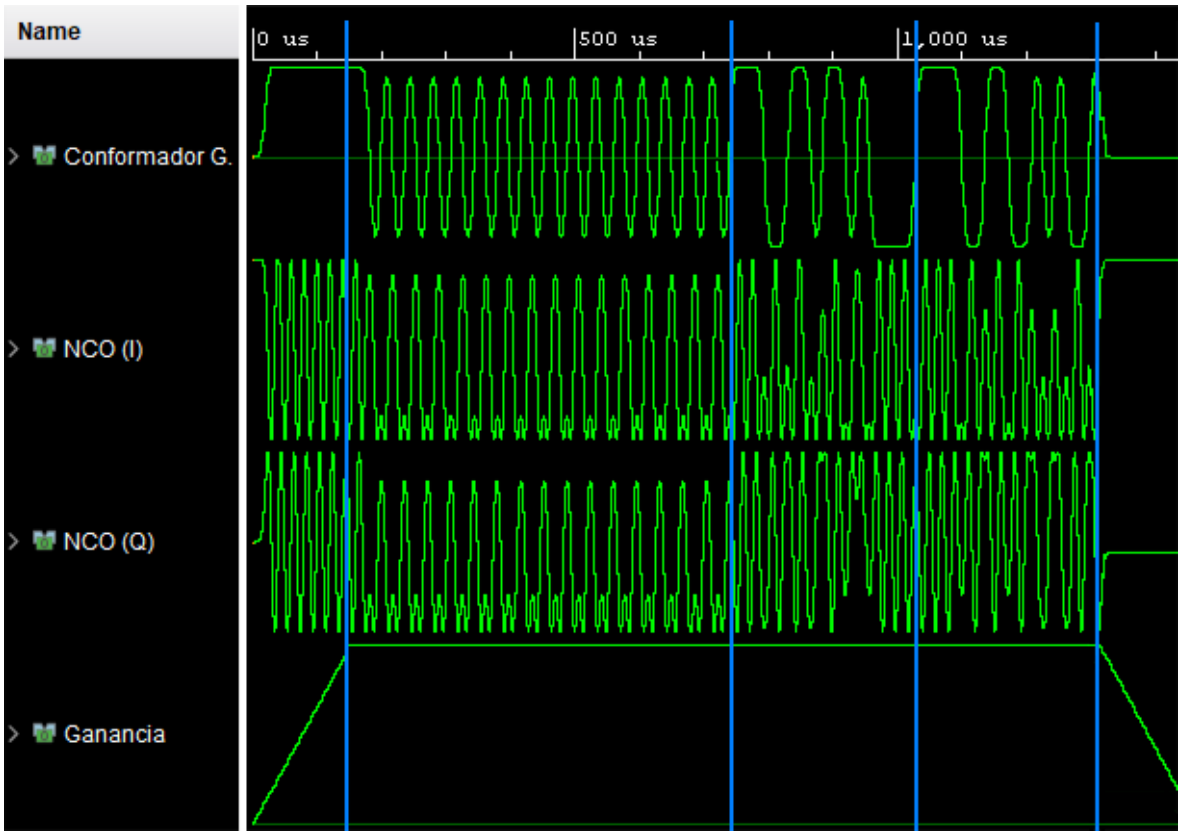


Figura 7.2: Co-simulación de la transmisión completa de un paquete con 2 B de datos. Con líneas azules se marca la separación de cada parte del paquete (Figura 3.3). La forma de onda *Conformador Gaussiano* representa la salida del conformador de pulsos gaussianos (Sección 5.4). Las formas de onda *NCO (I)* y *NCO (Q)* representan las salidas del NCO (Sección 5.5). La forma de onda *Ganancia* representa la salida del bloque empaquetador (Sección 5.3).

7.2. Resultados obtenidos del receptor sin ruido

La salida del transmisor fue ingresada directamente al receptor implementado y se analizó el comportamiento de sus elementos interactuando en conjunto. En la Figura 7.3 se observa la simulación obtenida de la demodulación, sincronización y decodificación.

Se puede ver como cada elemento posee el comportamiento esperado y el sistema responde correctamente. El PLL se sincroniza rápidamente con la señal de entrada y la demodula de manera aceptable. Sus salidas son las dos primeras señales de la Figura 7.3. Se observa que la forma de onda es muy similar a la transmitida (Figura 7.2).

El bloque Early-Late también logra una sincronización exitosa al cabo de algunos símbolos recibidos, cuyo ajuste realizado y frecuencia de decisión se ve en las señales *Ajuste fase* y *Decisión bit* respectivamente. Cuando la transmisión comienza, el sistema de referencia temporal del receptor se encuentra totalmente desfasado de la señal entrante. La señal *Ajuste fase* refleja como el bloque desplaza su referencia de tiempo, y luego de algunos símbolos recibidos la misma se estabiliza en cero. Esta oscilación impacta en la frecuencia de la señal *Decisión bit*, sin embargo estos cambios son en una escala de tiempo menor al de la imagen y no pueden diferenciarse fácilmente en la

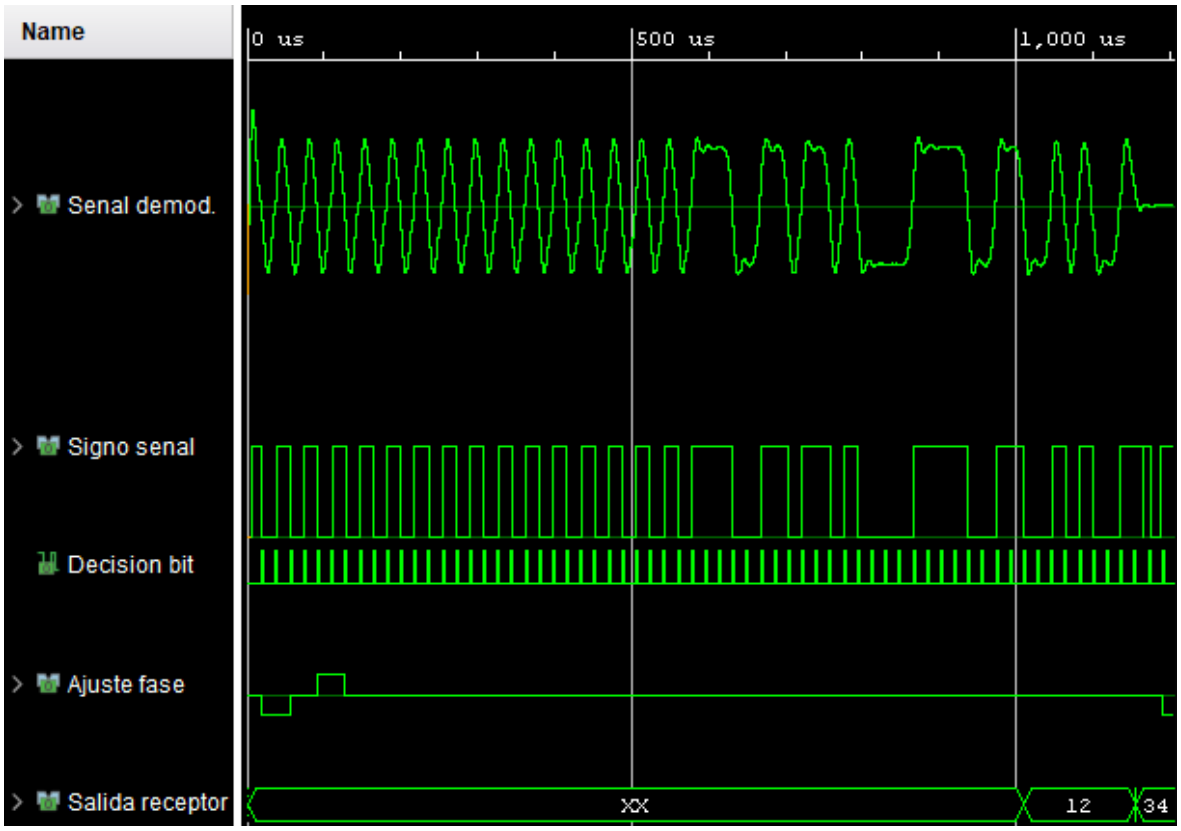


Figura 7.3: Co-simulación del sistema de recepción realizando la demodulación de un paquete completo. La forma de onda *Senal demod.* corresponde a la señal en banda base obtenida del PLL (Sección 6.4). La forma de onda *Senal senal* es la segunda salida del PLL que representa el signo de la señal demodulada y es entrada del siguiente módulo. *Decisión bit* simboliza el temporizado de la decisión sobre que bit fue transmitido del bloque Early-Late. *Ajuste fase* muestra el ajuste que realiza el bloque Early-Late para sincronizarse con el símbolo recibido. Por último *Salida receptor* es el resultado final de todo el sistema de recepción. La misma tiene una representación en bytes, por lo que en la figura se muestra su valor hexadecimal.

figura.

Por último, la salida del receptor completo (en bytes) está representada con la señal *Salida receptor* en la Figura 7.3. Los bytes obtenidos de todo el proceso de recepción fueron exactamente los mismos que se ingresaron en la cadena de transmisión en el apartado anterior.

7.3. Resultados obtenidos del receptor con ruido

El análisis realizado sobre desempeño del receptor se limitó a verificar su funcionamiento en presencia de ruido gaussiano blanco aditivo. Para evaluar el desempeño del sistema de forma más completa se debería realizar una transmisión de una gran cantidad de bits y verificar la cantidad de errores obtenidos. Sin embargo, cada una de estas simulaciones es muy costosa computacionalmente, por lo que los tiempo de simulación pueden llegar a ser muy elevados. Por este motivo se decidió dejar fuera del

alcance este tipo de validación más formal de la implementación.

Las pruebas realizadas cubrieron dos tipos de degradación de la señal. Por una parte se adicionó ruido gaussiano blanco aditivo, y por otra se aplicó una atenuación de la señal con respecto a la amplitud estacionaria del AGC (Sección 6.3). Estos efectos se aplicaron a la señal resultante de la co-simulación del transmisor completo (Sección 7.1).

Para este tipo de pruebas se utilizó el parámetro Relación Señal a Ruido SNR que consiste en la relación entre la potencia de la señal a la entrada del receptor y la potencia del ruido adicionado. Otro parámetro comúnmente utilizado es la relación entre la Energía de bit y la densidad espectral de potencia de ruido E_b/N_0. La relación que existe entre estos parámetros es la que se muestra en la Ecuación 7.1, donde T_s es el tiempo entre muestras y T_{sym} es el tiempo de símbolo.

$$\text{SNR} = \frac{E_b}{N_0}\frac{T_s}{T_{sym}} \tag{7.1}$$

Los valores de Relación Señal a Ruido SNR y atenuación A_{tt} utilizados para las pruebas fueron:

Prueba 1 : SNR $= 6\,\text{dB}$ y $A_{tt} = 0\,\text{dB}$.

Prueba 2 : SNR $= 6\,\text{dB}$ y $A_{tt} = 3\,\text{dB}$.

Prueba 3 : SNR $= 6\,\text{dB}$ y $A_{tt} = 10\,\text{dB}$.

Prueba 4 : SNR $= 0\,\text{dB}$ y $A_{tt} = 0\,\text{dB}$.

Prueba 5 : SNR $= 0\,\text{dB}$ y $A_{tt} = 3\,\text{dB}$.

Prueba 6 : SNR $= 0\,\text{dB}$ y $A_{tt} = 10\,\text{dB}$.

Prueba 7 : SNR $= -6\,\text{dB}$ y $A_{tt} = 0\,\text{dB}$.

Prueba 8 : SNR $= -6\,\text{dB}$ y $A_{tt} = 3\,\text{dB}$.

Prueba 9 : SNR $= -6\,\text{dB}$ y $A_{tt} = 10\,\text{dB}$.

La forma de onda *Senal demod.* corresponde a la señal en banda base obtenida del PLL (Sección 6.4). La forma de onda *Senal senal* es la segunda salida del PLL que representa el signo de la señal demodulada y es entrada del siguiente módulo. *Decisión bit* simboliza el temporizado de la decisión sobre que bit fue transmitido del bloque Early-Late. *Ajuste fase* muestra el ajuste que realiza el bloque Early-Late para sincronizarse con el símbolo recibido. El rango graficado va desde -4 a 4. Por último *Salida receptor* es el resultado final de todo el sistema de recepción. La misma tiene una representación en bytes, por lo que en las figuras se muestra su valor hexadecimal.

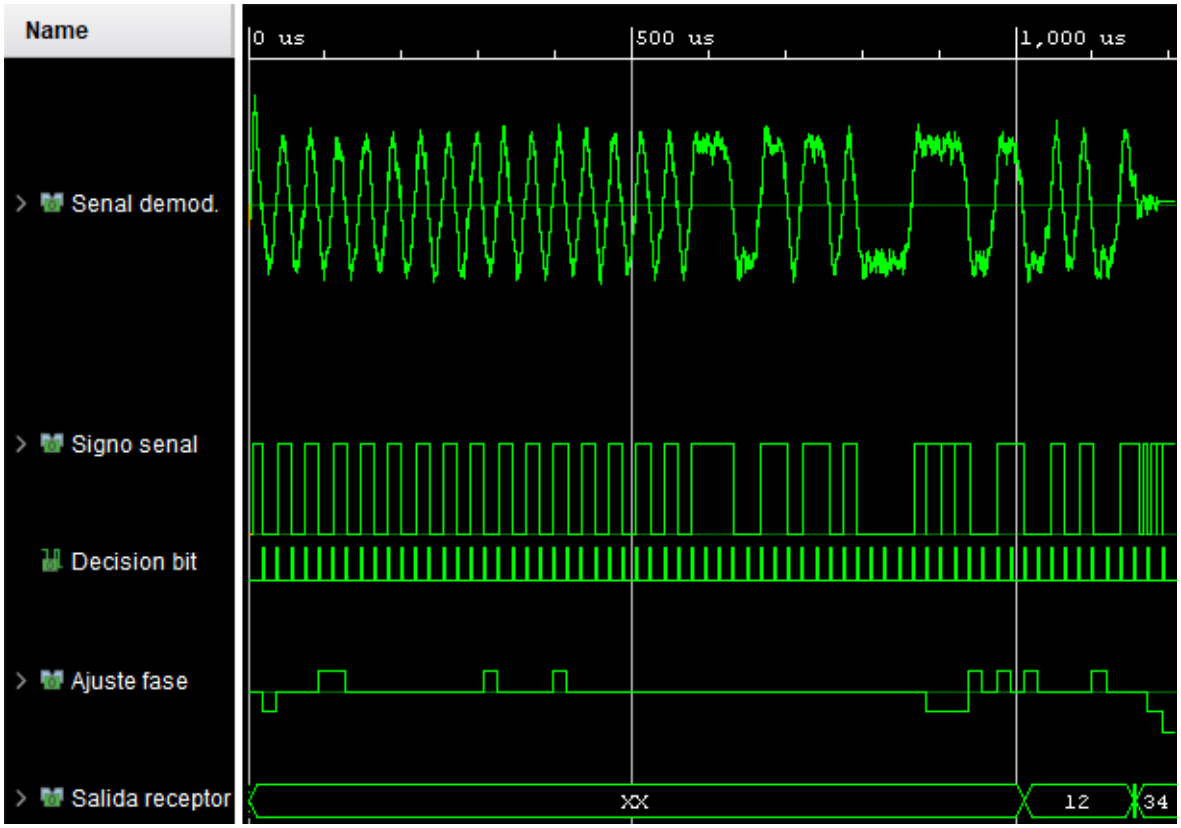


Figura 7.4: Resultado obtenido de la *Prueba 1*. Co-simulación del sistema de recepción con una SNR de 6dB sin atenuación realizando la demodulación de un paquete completo.

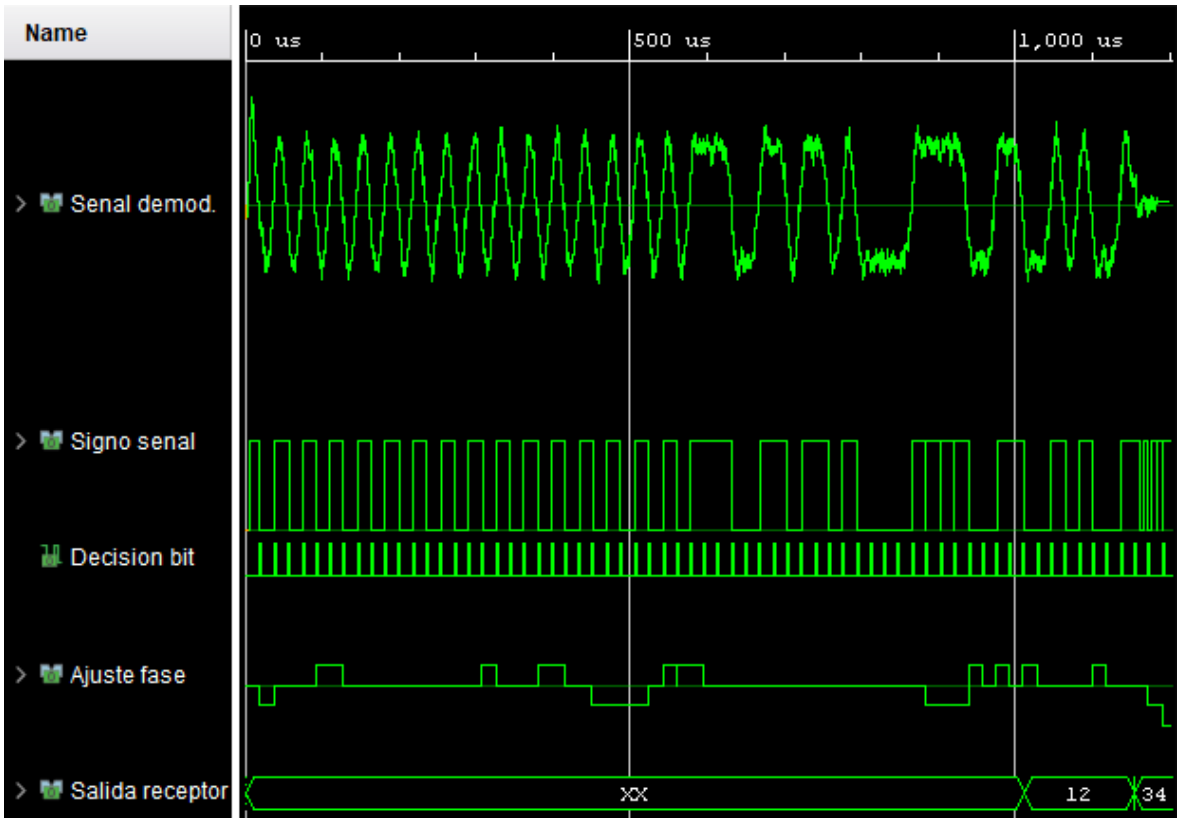


Figura 7.5: Resultado obtenido de la *Prueba 2*. Co-simulación del sistema de recepción con una SNR de 6dB y atenuación de 3dB realizando la demodulación de un paquete completo.

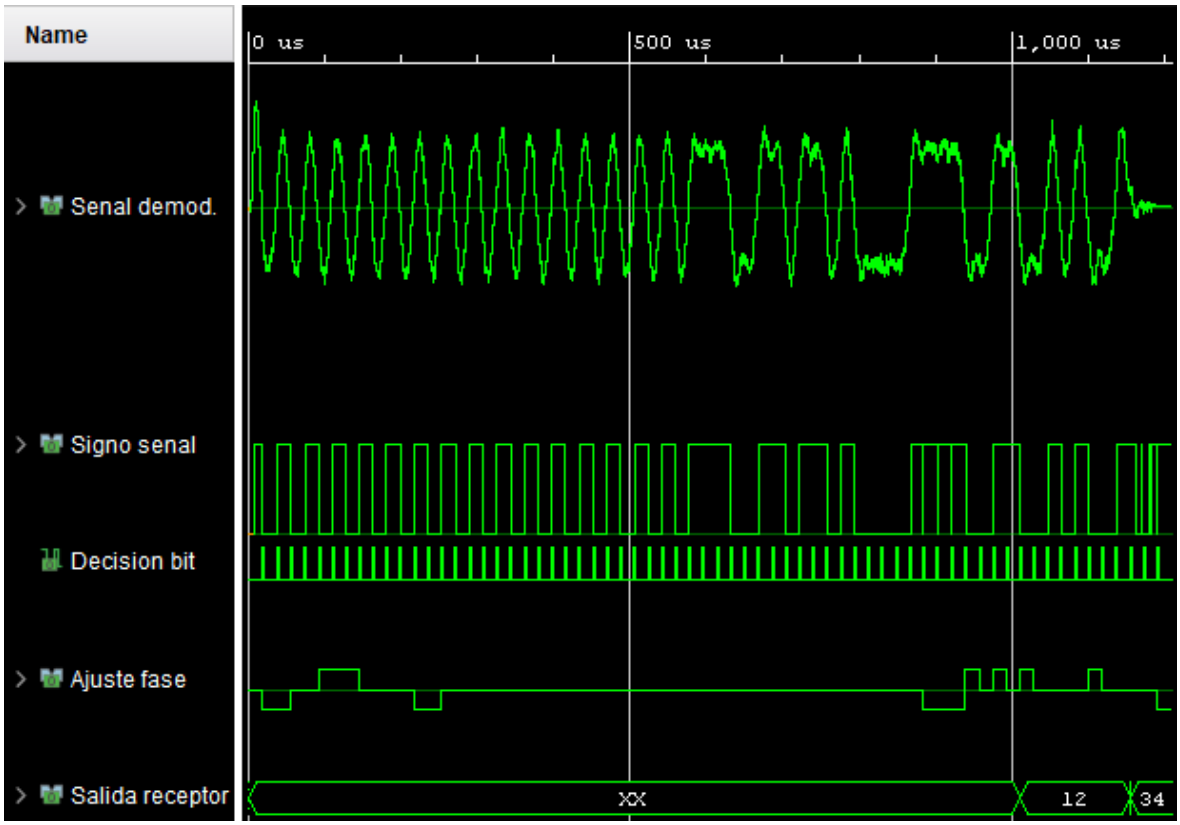


Figura 7.6: Resultado obtenido de la *Prueba 3*. Co-simulación del sistema de recepción con una SNR de 6dB y atenuación de 10dB realizando la demodulación de un paquete completo.

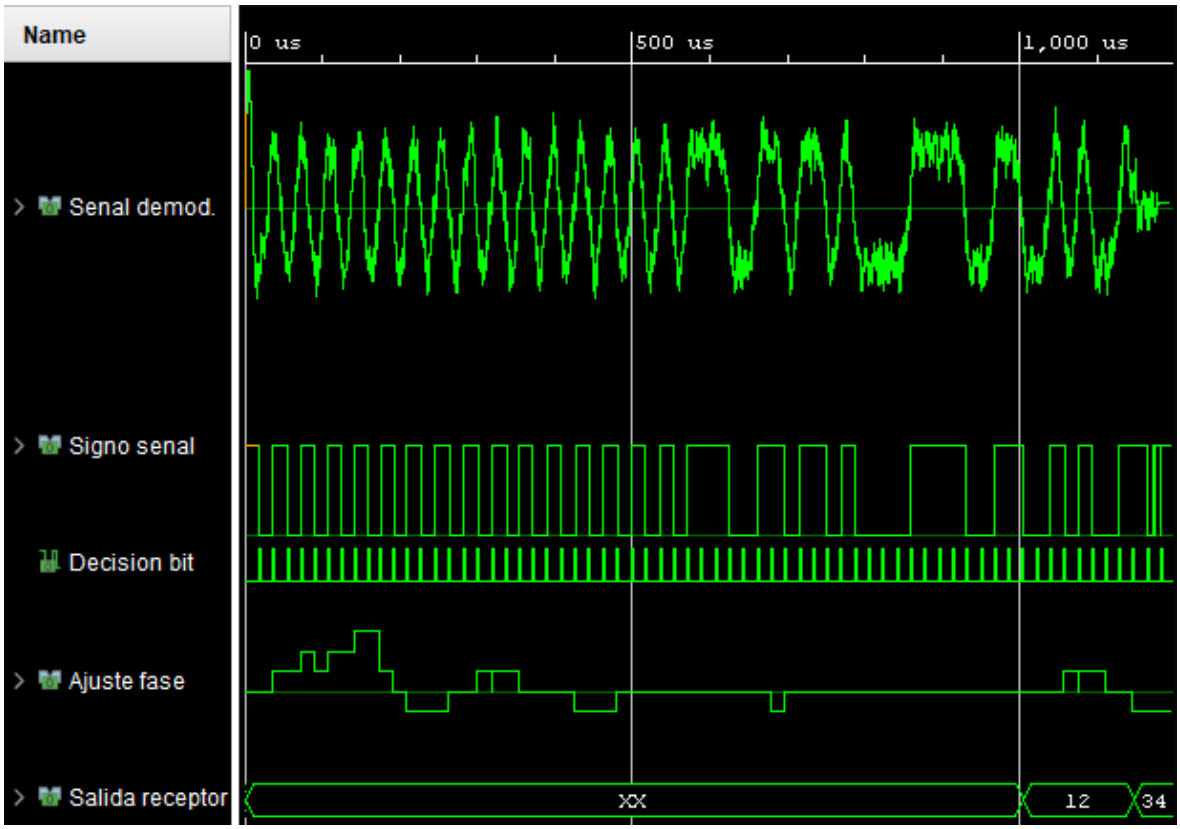


Figura 7.7: Resultado obtenido de la *Prueba 4*. Co-simulación del sistema de recepción con una SNR de 0dB sin atenuación realizando la demodulación de un paquete completo.

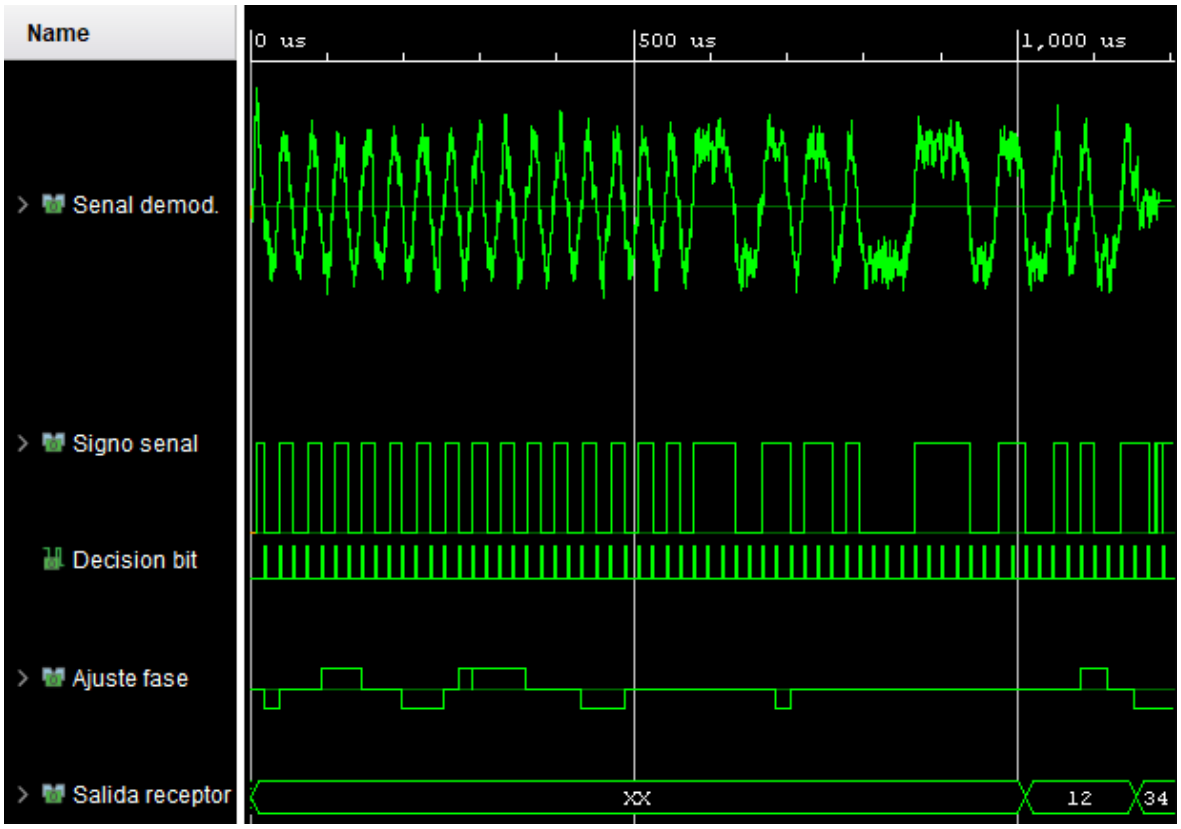


Figura 7.8: Resultado obtenido de la *Prueba 5*. Co-simulación del sistema de recepción con una SNR de y atenuación de 3dB realizando la demodulación de un paquete completo.

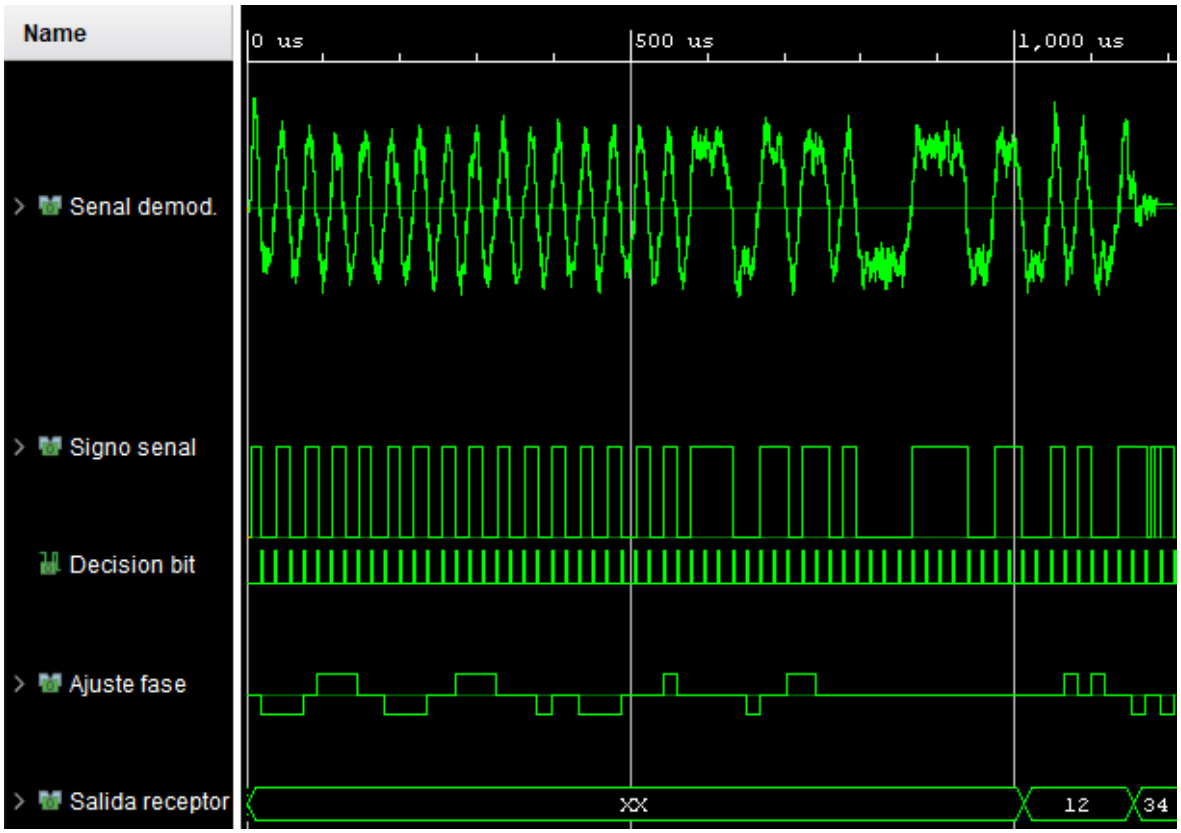


Figura 7.9: Resultado obtenido de la *Prueba 6*. Co-simulación del sistema de recepción con una SNR de 0dB y atenuación de 10dB realizando la demodulación de un paquete completo.

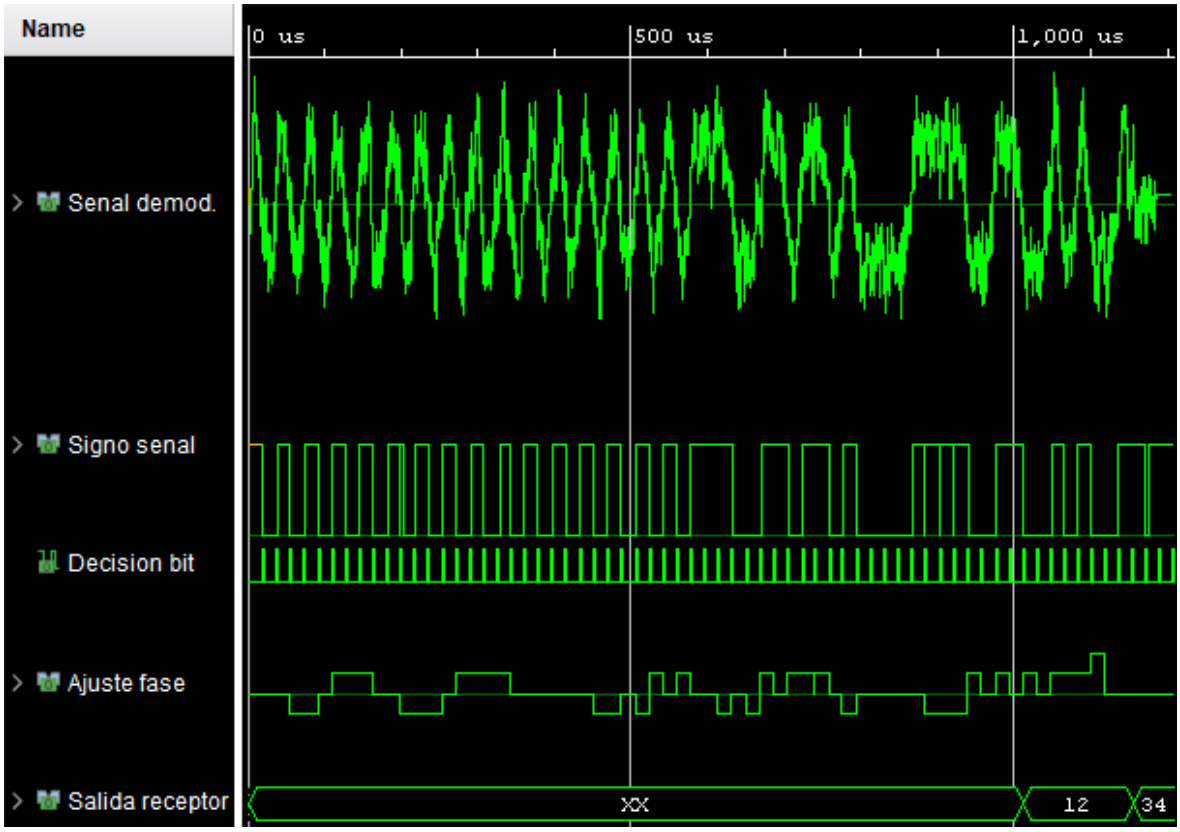


Figura 7.10: Resultado obtenido de la *Prueba 7*. Co-simulación del sistema de recepción con una SNR de -6dB sin atenuación realizando la demodulación de un paquete completo.

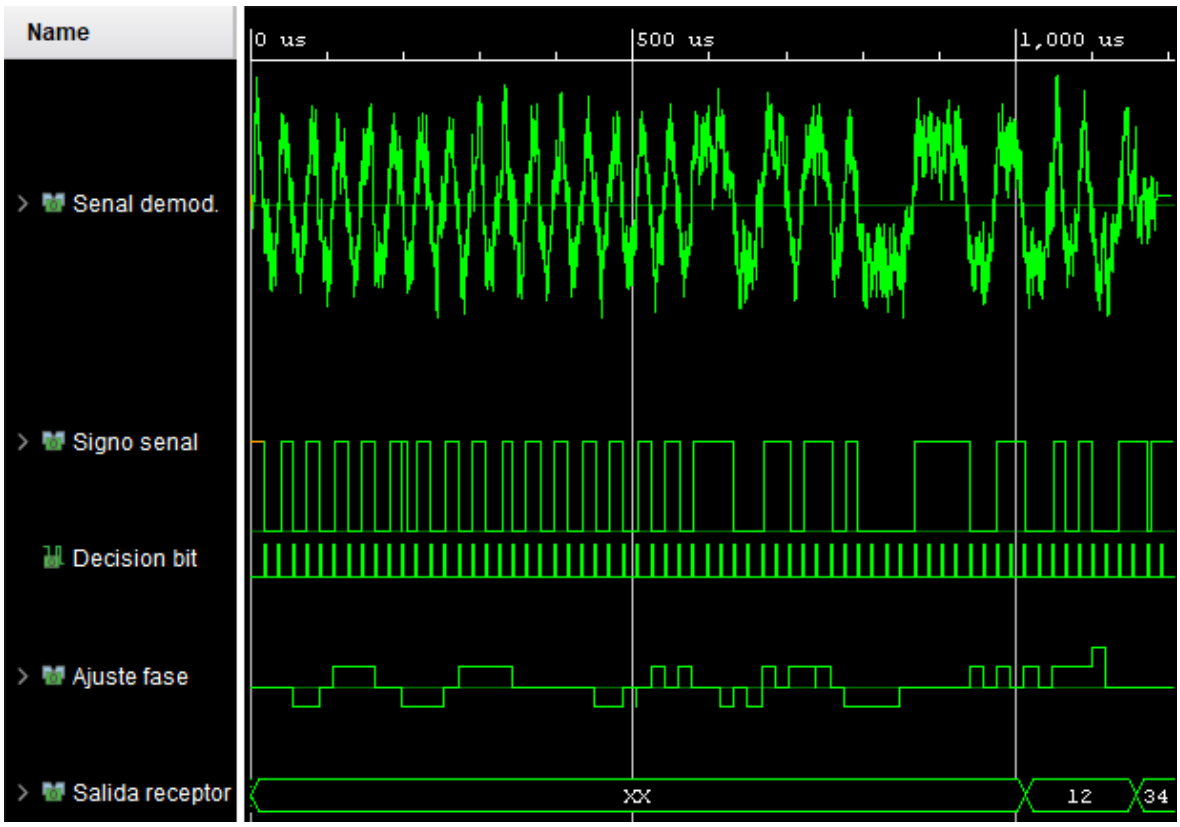


Figura 7.11: Resultado obtenido de la *Prueba 8*. Co-simulación del sistema de recepción con una SNR de -6dB y atenuación de 3dB realizando la demodulación de un paquete completo.

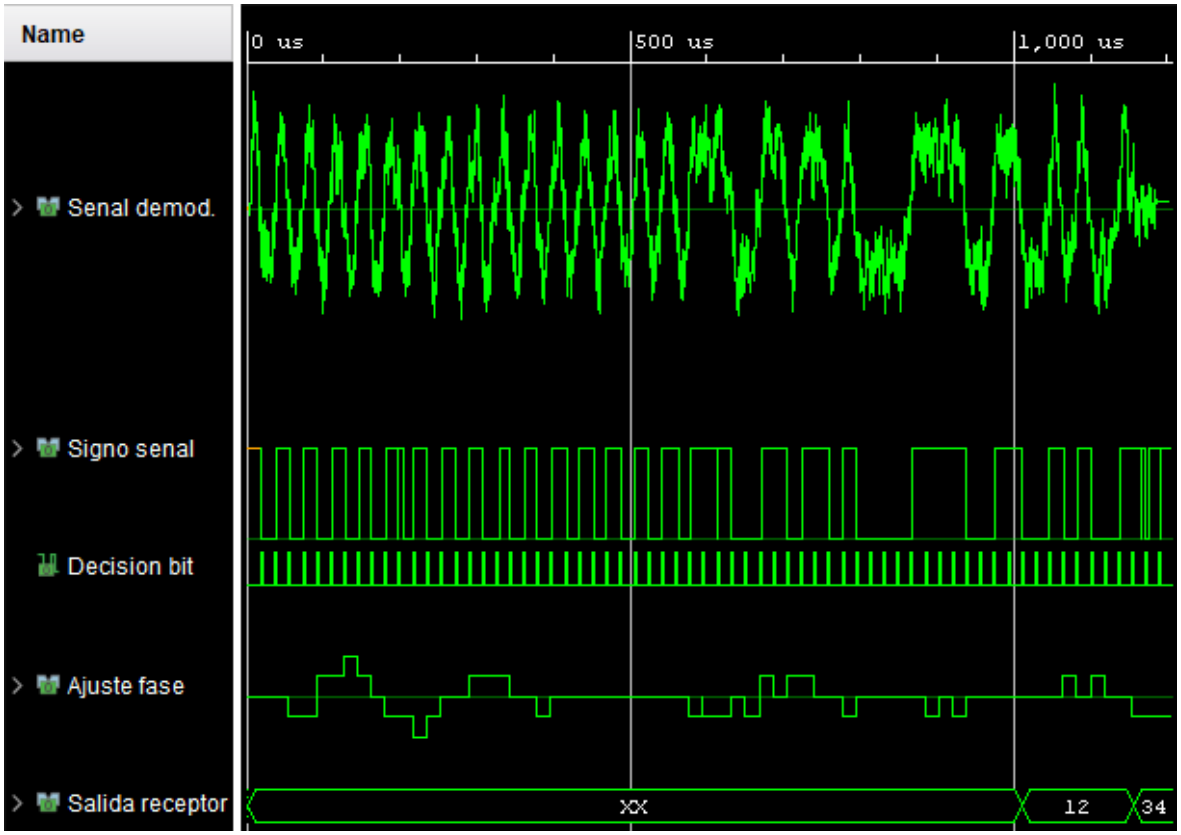


Figura 7.12: Resultado obtenido de la *Prueba 9*. Co-simulación del sistema de recepción con una SNR de -6dB y atenuación de 10dB realizando la demodulación de un paquete completo.

El primer aspecto para analizar es que en todas las instancias de prueba, el sistema diseñado logró demodular y decodificar exitosamente los datos que habían sido originalmente introducidos en la cadena de transmisión. Este resultado es de gran importancia ya que valida el funcionamiento de la cadena de recepción al menos para los rangos de SNR y atenuación probados.

Si se compara la co-simulación de la recepción sin ruido con respecto a la recepción con ruido, se puede observar una notable degradación de la señal demodulada. A medida que la SNR disminuye, la degradación es cada vez más mayor. Esta oscilación ruidosa sobre el símbolo demodulado podría ser disminuida utilizando filtros más ajustados y con transiciones más abruptas entre la banda de paso y la de rechazo. Esta mejora es a costa de mayor requerimiento de recursos. Otra opción posible sería utilizar filtros con respuesta al impulso infinita (filtros IIR). Llegado el caso en que esta oscilación cruce el eje hacia los valores negativos, se podrían generar muestras contrarias al símbolo e inducir errores. Este efecto sería mucho menor si la detección de símbolo fuera utilizando un filtro óptimo.

De igual forma, en ningún caso el ruido adicionado fue lo suficientemente grande para generar un error de bit. Esto era esperable debido a que la cantidad de bits utilizados durante las pruebas es pequeña y la SNR no es lo suficientemente baja. Para realizar una prueba más representativa y estimar la tasa de error de bit se deberían haber transmitido una cantidad de bits considerablemente mayor.

Se observa además que el ajuste de fase no llega a estabilizarse en ningún momento. Esto se debe a que las transiciones son ruidosas y generan variaciones en los cálculos que realiza el módulo de Early-Late. Por otro lado, se puede ver que si bien el ajuste no se estabiliza completamente, la oscilación se mantiene alrededor de cero. Es decir que cada error en el ajuste de fase inducido por el ruido es compensado en las siguientes muestras.

Por último se puede ver que la atenuación introducida no es un problema para el sistema. Este comportamiento se debe a la utilización del módulo AGC, el cual amplifica la señal de entrada como se explicó en la Sección 6.3. Esta característica es deseable en un receptor inalámbrico debido a los cambios abruptos de amplitud de la señal de entrada ocasionados por el canal.

Capítulo 8

Conclusiones

Se realizó el análisis de protocolos al inicio del proyecto y se descubrió la gran variedad de tecnologías disponibles para IoT. Cada opción posee un conjunto de ventajas y desventajas particulares. A la hora de llevar a cabo una implementación todas estas variables tienen que ser evaluadas de alguna forma. Existen protocolos muy utilizados en el mercado debido a su buen desempeño pero que sin embargo poseen especificaciones cerradas. Otros por el contrario poseen especificaciones abiertas pero son muy poco utilizados, existe poca documentación sobre su implementación y la comunidad que lo soporta es pequeña, como es el caso de D7A. Este panorama influyó mucho a la hora de elegir un protocolo para implementar en el proyecto ya que limitó en gran medida las opciones disponibles.

El análisis de factibilidad fue exitoso en la medida que se valoraron múltiples aspectos del proyecto y se enfrentó el desafío tomando en cuenta los recaudos necesarios.

En cuanto al diseño del transmisor y del receptor, se logró obtener un modelo funcional que se ajusta a las especificaciones de D7A. Este modelo no es óptimo y se mencionó en varias oportunidades, pero durante la implementación se trató de no dejar de lado el costo en recursos que puede requerir una solución mejor. Estas decisiones realizadas tomando en cuenta la relación de compromiso que existía en cada caso, son ejemplo del trabajo de ingeniería que se requiere para un proyecto de estas magnitudes. Estas determinaciones no fueron evaluadas de manera completa y detallada, sin embargo haber tenido en cuenta las diferentes opciones formó parte de un aprendizaje valioso.

Alineado con lo anterior, se intentó abarcar la mayor cantidad de funcionalidad posible dentro del proyecto sacrificando optimalidad en cada bloque. Se priorizó obtener un resultado completo aunque el desempeño no sea ideal. Para lograr esto se redujo la cantidad de pruebas de cada módulo. Por otro lado se obtuvieron resultados aceptables muy rápidamente.

El flujo de trabajo utilizando HLS resultó muy complicado al principio del proyecto.

Se tuvo una curva de aprendizaje más lenta de lo esperado debido a que la documentación que existe de la metodología es relativamente escasa. Sin embargo, una vez que se entendieron los conceptos más relevantes del proceso, el desarrollo de funcionalidades se tornó ágil.

Otro de los inconvenientes que se enfrentó fue la transparencia con la que HLS implementa el programa de alto nivel. En principio esto no debiera ser un problema, sino que por el contrario es el objetivo de la herramienta. Pero para este tipo de desarrollo resulta importante conocer los detalles de bajo nivel de la implementación. De cierta forma se logró ajustar ciertos parámetros de bajo nivel para satisfacer los requerimientos más fundamentales, pero resultó muy complicado optimizar adecuadamente el desarrollo. Este proceso requería mucho tiempo para evaluar el desempeño resultante con cada variante de la especificación.

Un aspecto positivo de esta metodología es que a medida que la funcionalidad requerida era de más alto nivel, el tiempo necesario para la programación y las pruebas se reducía notablemente. Resultó impactante la facilidad con la que se podían realizar cambios en la funcionalidad de un módulo. Esta resulta ser una de las grandes ventajas de la herramienta y se pudo verificar durante el proyecto.

Otro de los beneficios de HLS es la reutilización de código de alto nivel. El caso más paradigmático dentro del proyecto fue la utilización de código de terceros en la implementación del decodificador Viterbi. En un principio, esta parte de la decodificación iba a ser dejada de lado debido al tiempo que demandaría desarrollar y depurar un algoritmo de esa complejidad. Sin embargo, exportar de forma casi directa un algoritmo ya desarrollado de estas características a un lenguaje de descripción de hardware es un resultado no menor.

8.1. Propuestas de trabajos futuros

El presente proyecto se puede extender y complementar de muchas maneras. En primer lugar se podría realizar la optimización de cada bloque y una revisión detallada de los recursos que requiere el desarrollo. Para esto se debería hacer un análisis mucho más profundo del impacto que tiene cada implementación y de las relaciones de compromiso en cada caso. Es importante mencionar que se requeriría un conocimiento avanzado de HLS para continuar por este camino.

Otro tipo de trabajo vinculado sería la realización de la etapa de radiofrecuencia. Es decir, a partir de la señal en frecuencia intermedia, se podría realizar toda la etapa de amplificación, mezclado y filtrado para ser transmitido por la antena. Paralelamente, se podría diseñar el Front-End de radiofrecuencia para la recepción de la señal y la bajada a frecuencia intermedia.

Como última sugerencia se propone implementar la capa superior del protocolo. En

otras palabras, continuar con la implementación del acceso al medio definido en la especificación. Esta alternativa se considera bastante desafiante si se decide implementar utilizando HLS debido a que el temporizado debe ser ajustado de manera muy configurable y de forma precisa. Sin embargo la experiencia puede resultar muy enriquecedora.

Bibliografía

[1] 3rd Generation Partnership Project (3GPP). Standards for the IoT. URL `http://www.3gpp.org/news-events/3gpp-news/1805-iot_r14`. 1, 12

[2] LoRa Alliance. LoRa Alliance Oficial Website. URL `https://www.lora-alliance.org/what-is-lora`. 1, 13

[3] Sigfox. Sigfox Oficial Website. URL `https://www.sigfox.com/en/sigfox-iot-technology-overview`. 1, 13

[4] Coussy, P., Gajski, D. D., Meredith, M., Takach, A. An introduction to high-level synthesis. *IEEE Design & Test of Computers*, **26** (4), 8–17, 2009. 2, 4

[5] Ashton, K., *et al.* That 'internet of things' thing. *RFID journal*, **22** (7), 97–114, 2009. 5

[6] UIT-T Comisión de Estudio 13. Descripción general de internet de los objetos. Recomendación Y.2060, Unión Internacional de Telecomunicaciones, jun. 2012. 5, 6

[7] Reglamento de radiocomunicaciones. Inf. téc., Unión Internacional de Telecomunicaciones, 2016. 9, 17

[8] ENACOM. Dispositivos de baja potencia. Norma Técnica Q2-60.14 V17.1, Ministerio de Modernización, jul. 2017. 10, 26, 27

[9] Mahmoud, M. S., Mohamad, A. A. A study of efficient power consumption wireless communication techniques/modules for internet of things (iot) applications. *Advances in Internet of Things*, **6** (02), 19, 2016. 10

[10] van Bussel, J. Overview of iot connectivity and protocols - internet of things. *Whitepaper*, 2016. 11

[11] Weightless SIG. Weightless SIG Oficial Website. URL `http://www.weightless.org/about/what-is-weightless`. 13

[12] DASH7 Alliance. DASH7 Alliance Oficial Website. URL `http://www.dash7-alliance.org/why-dash7/`. 14

[13] DASH7 Alliance Wireless Sensor and Actuator Network Protocol. Specification V1.1, DASH7 Alliance, 2017. 17

[14] CNC. Dispositivos de baja potencia. Resolución 226/2008, Secretaría de comunicaciones, 2008. 26

[15] ENACOM. Requisitos de ensayos para equipos de banda ancha para uso privado. Norma Técnica 14/2013 V13.2, Ministerio de Modernización, sep. 2013. 26

[16] Xilinx. AXI. Reference Guide UG761, Xilinx Inc., mar. 2011. 31, 33, 37, 43, 45, 49, 53, 54, 64, 67, 71, 72, 75, 76

[17] Simon, H. Digital Communication Systems, tomo 21. Wiley, 2014. 37

[18] Proakis, J. G., Salehi, M., Zhou, N., Li, X. Communication systems engineering, tomo 2. Prentice Hall New Jersey, 1994. 48, 52, 60

[19] Sklar, B. Digital communications, tomo 2. Prentice Hall Upper Saddle River, 2001. 53, 66, 71

[20] Smith, S. W., *et al.* The scientist and engineer's guide to digital signal processing, 1997. 60, 61

[21] Rice, M. Digital Communications: A Discrete-Time Approach. Prentice Hall, 2008. 61

[22] University of Antwerp. OSS-7: Open Source Stack for Dash7 Alliance Protocol. URL `https://github.com/MOSAIC-LoPoW/dash7-ap-open-source-stack`. 72